



US006563919B1

(12) **United States Patent**  
**Aravamudhan et al.**

(10) **Patent No.:** **US 6,563,919 B1**  
**(45) Date of Patent:** **May 13, 2003**

(54) **SYSTEM AND METHOD FOR UNIFYING THE IMPLEMENTATION AND PROCESSING OF MOBILE COMMUNICATIONS AND A UNIFIED MOBILITY MANAGER FOR PROVIDING SUCH COMMUNICATIONS**

# FOREIGN PATENT DOCUMENTS

DE 198 01 563 A1 7/1999  
 WO WO 98/20647 5/1998

# OTHER PUBLICATIONS

*Universal Protocol Conversion*, IBM Technical Disclosure Bulletin, Dec. 1995, vol. 38, No. 12.

\* cited by examiner

*Primary Examiner*—Nay Maung  
*Assistant Examiner*—Tanmay Lele

(74) *Attorney, Agent, or Firm*—Paul W. Fulbright; John D. Crane; Bracewell & Patterson, L.L.P.

(75) **Inventors:** **Lakshminarasimhan Aravamudhan**, Plano, TX (US); **John P. Larkins**, Plano, TX (US); **Rohit Gupta**, Plano, TX (US); **Haitao Li**, Plano, TX (US)

(73) **Assignee:** **Nortel Networks Limited**, St. Laurent (CA)

(\*) **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) **Appl. No.:** **09/465,177**

(22) **Filed:** **Dec. 17, 1999**

(51) **Int. Cl.**<sup>7</sup> ..... **H04M 7/00**

(52) **U.S. Cl.** ..... **379/230; 379/229; 455/426; 455/435; 370/401; 370/466**

(58) **Field of Search** ..... **455/426, 435, 455/433, 460; 379/229, 230; 370/338, 328, 352, 401, 466**

# (56) References Cited

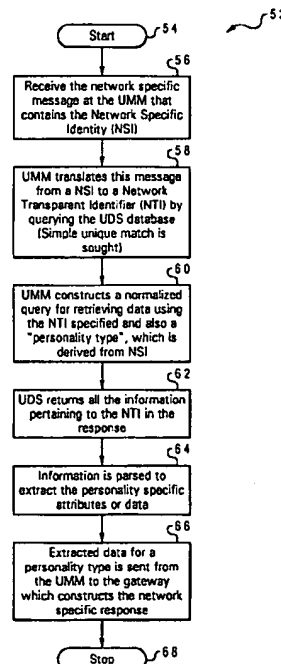
## U.S. PATENT DOCUMENTS

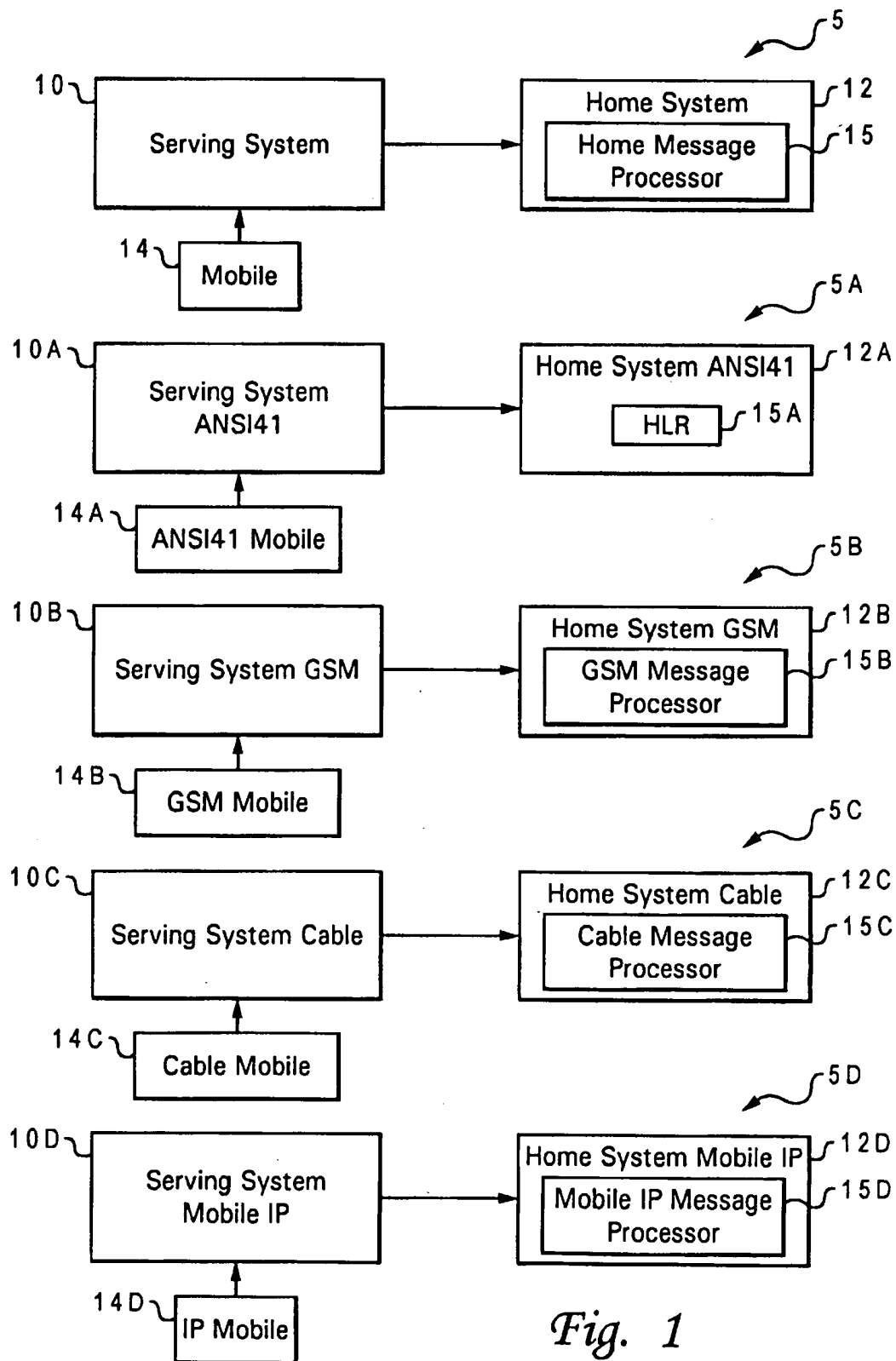
5,852,660 A \* 12/1998 Lindquist et al. .... 379/230  
 5,966,431 A \* 10/1999 Rieman et al. .... 379/115  
 6,393,112 B1 \* 5/2002 Gottlieb et al. .... 379/112.01  
 6,418,306 B1 \* 7/2002 McConnell ..... 455/413

# (57) ABSTRACT

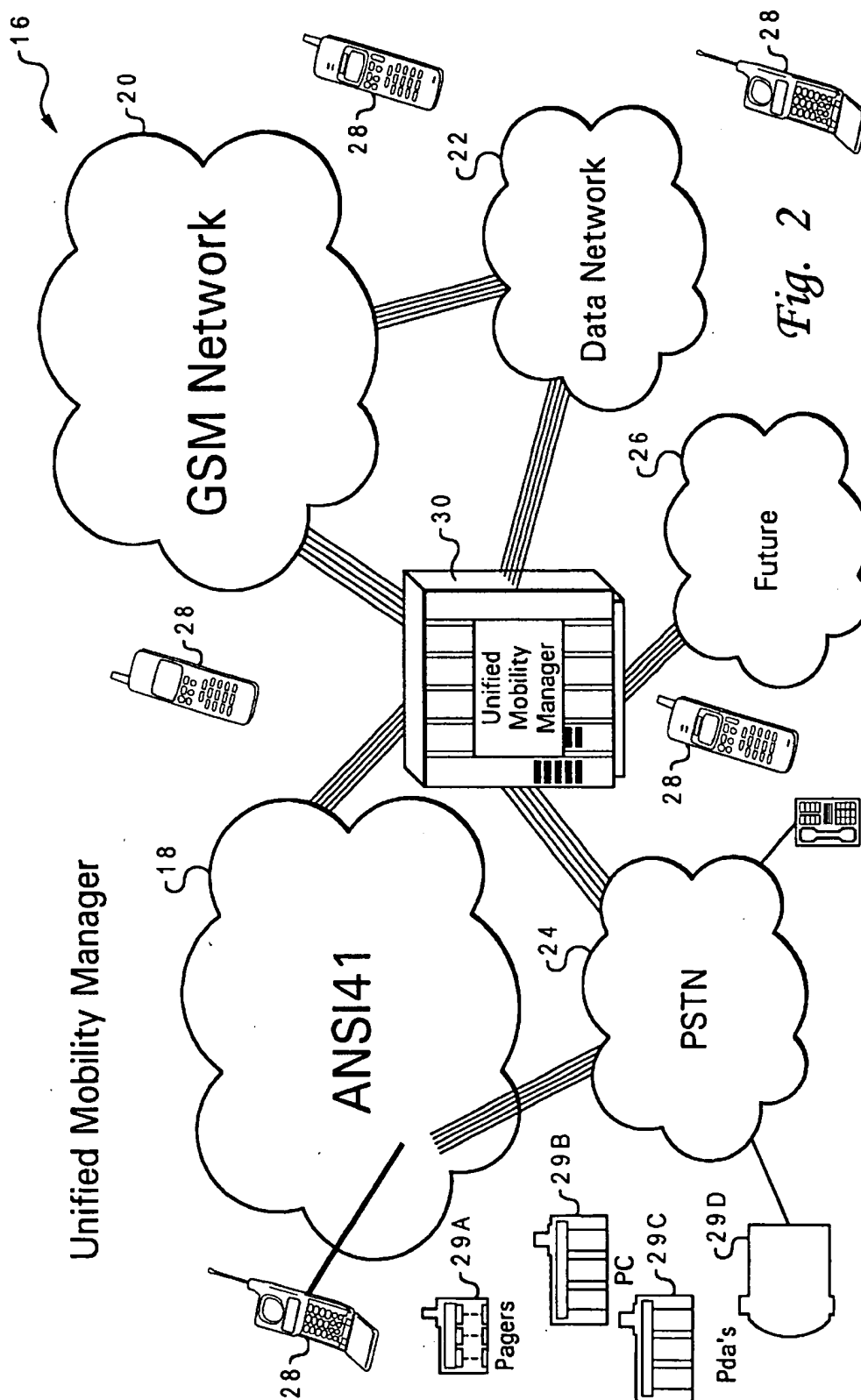
A gateway cluster has a number of gateways for different types of communication protocols. The respective gateways convert network messages to normalized messages by querying the categories, data, and network types of the normalized data for the mobile systems from which the network messages were generated. A database system stores normalized data in categories. The normalized data include data relating to the mobile systems and network types for the data. A unified mobility manager is coupled to and in communications with the gateway cluster and the database system. The unified mobility manager receives and processes the normalized messages, performs operations based on the normalized messages and on the categories, the data, and the network types of the normalized data, and formulates normalized responses responsive to the normalized messages. The normalized responses are converted to network responses at the gateways, and the network responses are sent to the respective mobile systems.

**22 Claims, 7 Drawing Sheets**





*Fig. 1*  
*Prior Art*



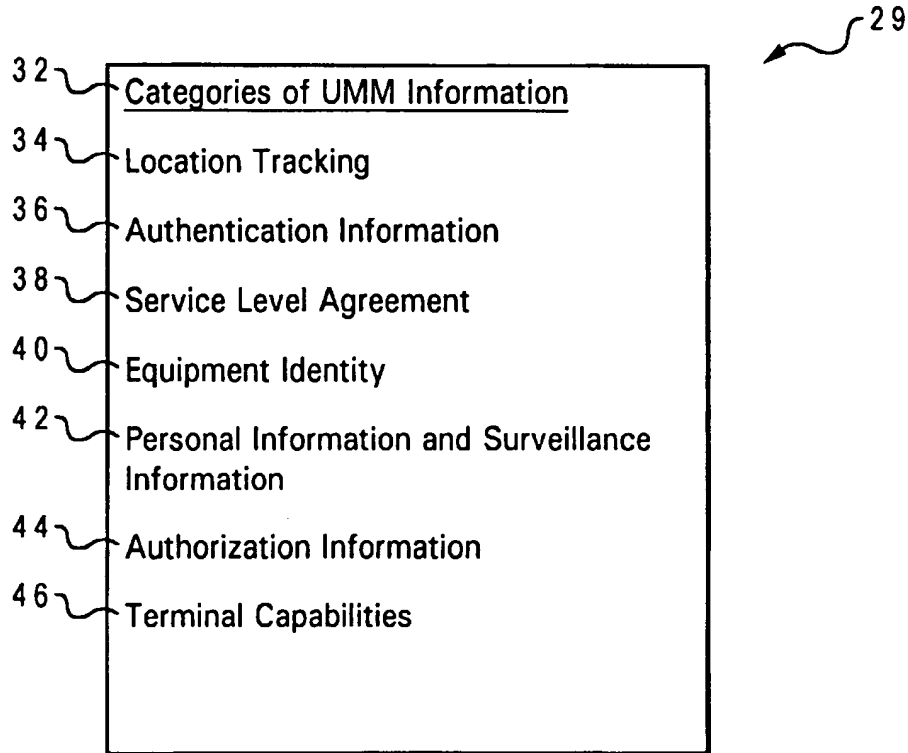
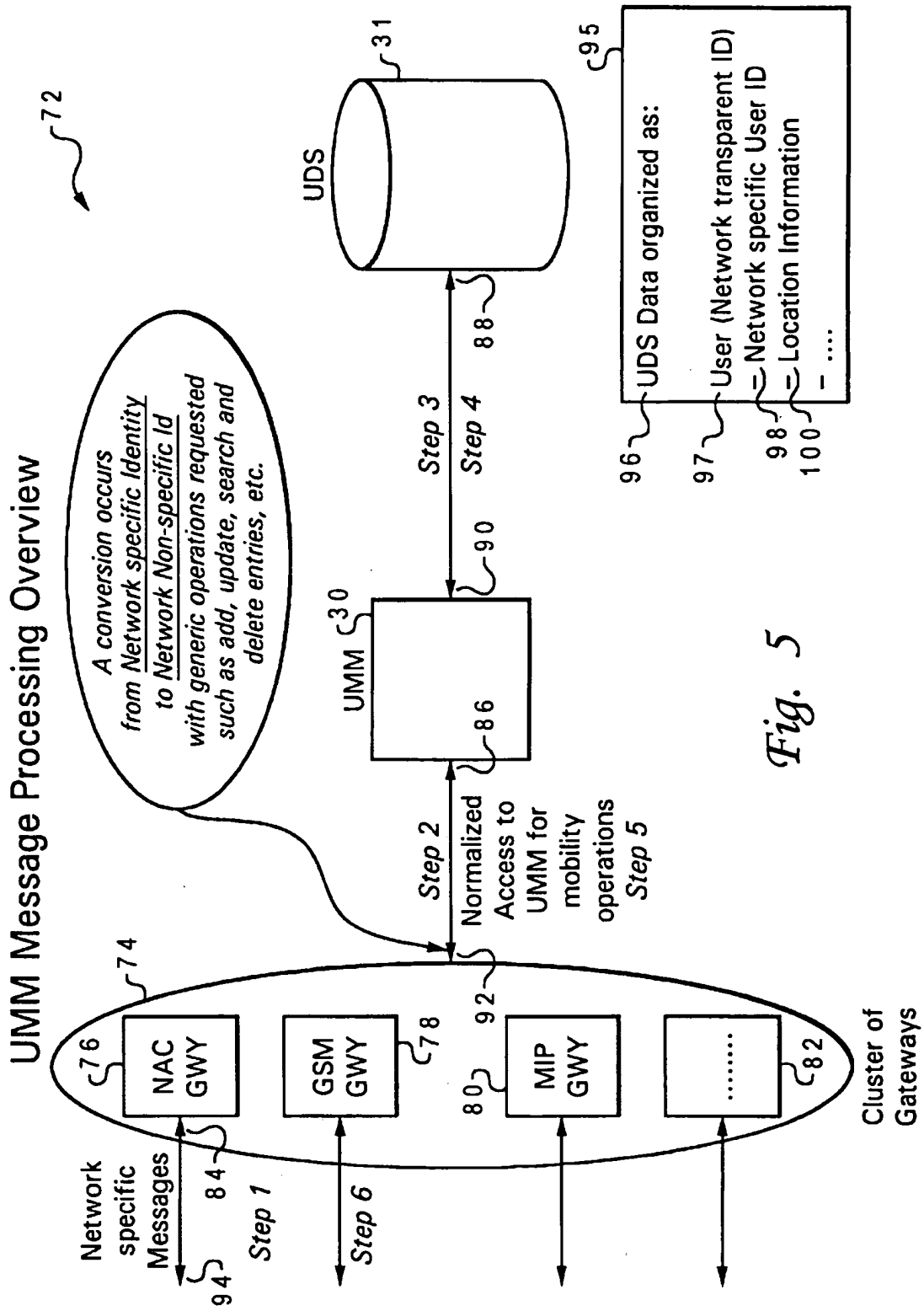
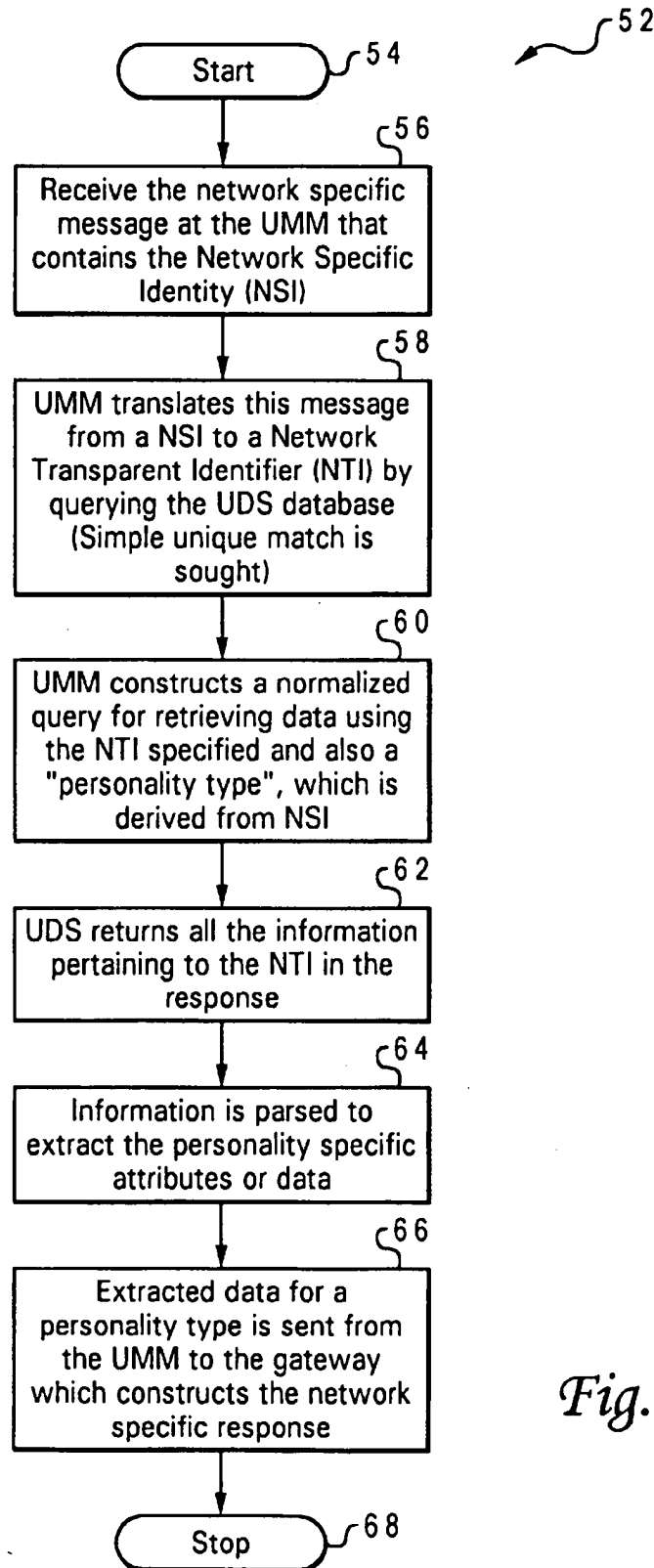
*Fig. 3*

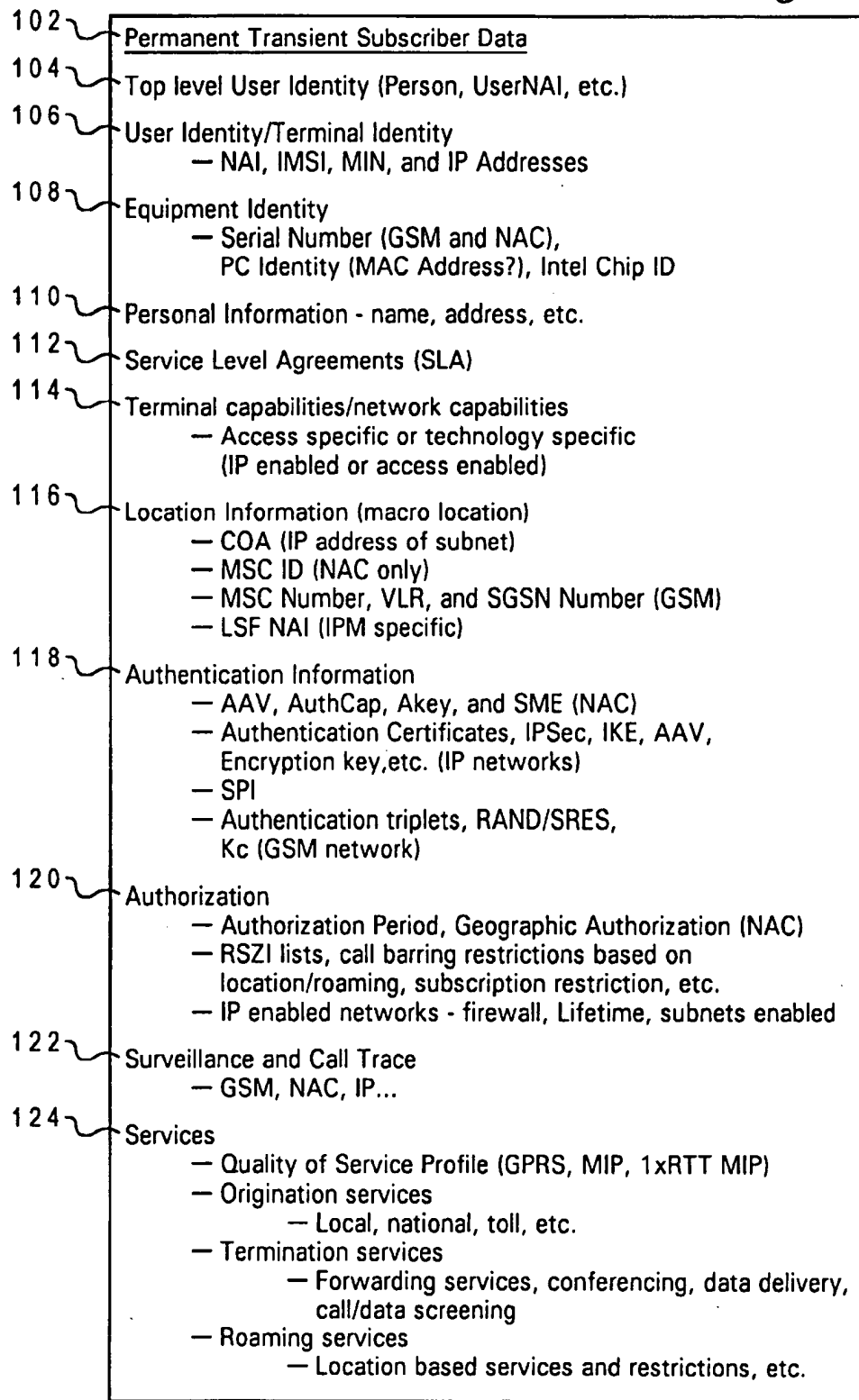
Diagram illustrating Authentication Information (36) (47). The information is organized into a table (50) with two columns: Network/Personality and Data.

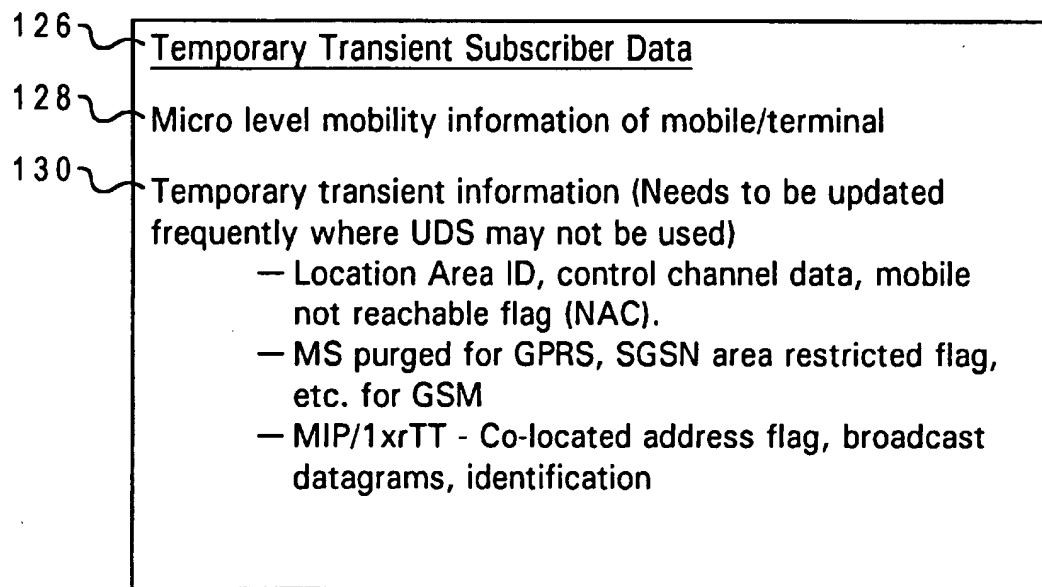
48 Network/ Personality	50 Data
18 ANSI41	AAV, AuthCap, Akey & SME (NAC)
Cable	Cable Modem specific Auth.
20 GSM GPRS	Auth. triplets, RAND/SRES and Kc
Mobile IP	Auth. Certificates, Encryption key

*Fig. 4*



*Fig. 6*

*Fig. 7*

*Fig. 8*

1

# SYSTEM AND METHOD FOR UNIFYING THE IMPLEMENTATION AND PROCESSING OF MOBILE COMMUNICATIONS AND A UNIFIED MOBILITY MANAGER FOR PROVIDING SUCH COMMUNICATIONS

## BACKGROUND OF THE INVENTION

### 1. Technical Field

The present invention relates in general to a system and method for unifying the implementation and processing of mobile communications by various mobile devices and units and terminals that operate on various mobile communication protocols and in particular to a unified mobility manager system and method that is used to provide such unified implementation and processing of mobile communications by various mobile devices and units that operate on various mobile communication protocols. Still more particularly, the present invention relates to normalizing messages from the various mobile devices and units and terminals, to processing, handling, and responding to these normalized messages, and to converting the normalized responses back to respective communication protocols.

### 2. Description of the Related Art

For the present description, the term "mobile" or "mobility" is used to define and describe a device, unit, terminal, or system that is able to be moved from one location to another location, whether the device, unit, terminal, or system is a wireless system that is movable or a wireline system that is movable. Furthermore, for this present description, the phrase "mobile communication" or "mobility communication" is used to define and describe communication protocols that track movement of such "mobile" devices, units, terminals, or systems from one location to another location and that provide and support communications for such "mobile" devices, units, terminals, or systems (i.e. whether these devices, units, terminals, or systems are wireless systems that are movable or wireline systems that are movable). As will be seen later in the specification, the present invention is not in any way limited to any particular communication device, unit, terminal, system, or respective communication protocol, and the present invention may be utilized with any device, unit, terminal, or system that is able to be moved from one location to another location (whether a wireless system or a wireline system) and with any communication protocol that tracks, provides, and supports communications for such devices, units, terminals, or systems from one location to another location.

In the telecommunications field, various access technologies and communication protocols exist. Examples of such access technologies and communication protocols are ANSI41 (North American Cellular), Group Special Mobile (GSM) network, cable modem network, mobile Internet protocol (IP), Public Switched Telephone Network (PSTN), data network, IS-136 Time Division Multiple Access (TDMA), IS-54B Time Division Multiple Access (TDMA), Advanced Mobile Phone System (AMPS), Code Division Multiple Access (CDMA), and any other such technologies that provide voice and data applications to a mobile customer.

With reference now to the figures and in particular with reference to prior art FIG. 1, FIG. 1 illustrates that each of a number of conventional communication systems employing differing technologies and protocols requires its own serving system 10 for a mobile unit or device or terminal 14 to locate and find its own home system 12 in order to enable

2

further communication based services to be provided to the mobile unit or device or terminal 14 (i.e. voice related applications, data related applications, or any other form of information exchange). A home system 12 is typically where the subscriber or user information resides, and a serving system 10 is any system that provides services to and that registers or attempts registration of a mobile unit or device or terminal 14 that is outside of its home system 12 by finding and forwarding a request(s) to the home system 12. FIG. 1 shows that the mobile system 14 is away from its home system 12 and is at a location in which it sends its request to the serving system 10. The serving system 10 then tracks, finds, and forwards the request to the home system 12 of the mobile system 14. The home system 12 has a home message processor 15 that processes the request. A response to the request is sent from the home message processor 15 to the mobile system 14 through the serving system 10.

FIG. 1 shows that each different type of mobile communication protocol involves a mobile unit or device or terminal type that requires and accesses its respective type of serving system and home system. For example, ANSI41 or North American Cellular (NAC) is the communication protocol for cellular communications in North America. FIG. 1 shows that if an ANSI41 mobile unit or device 14A (i.e. cellular telephone or device) is away from its home area (i.e. home system 12A), then the ANSI41 mobile unit or device 14A must communicate with a local and compatible ANSI41 serving system 10A. The local and compatible ANSI41 serving system 10A, in turn, finds and locates the home system 12A for that mobile unit or device 14A, which is an ANSI41 type, for the mobile unit or device 14A, and the serving system 10A sends a request(s) on behalf of the ANSI41 mobile unit or device 14A to its home system 12A. The home system 12A has an ANSI41 message processor or HLR 15A that is able to handle and process the ANSI41 request and sends back a corresponding response to the request.

FIG. 1 shows similar topologies for the GSM protocol, cable protocol, and Internet protocol (IP). For example, a GSM mobile unit or device 14B requires and is enabled with communications only through a GSM serving system 10B and a GSM home system 12B having a GSM message processor 15B that handles and processes GSM requests or messages. A cable modem mobile unit or device 14C similarly requires and is enabled with communications only through a cable modem serving system 10C and a cable modem home system 12C having a cable modem message processor 15C that handles and processes cable modem requests or messages, and an IP mobile unit or device 14D requires and is enabled with communications only through an IP serving system 10D and an IP home system 12D having an IP message processor 15D that handles and processes IP requests or messages.

Thus, each individual type of communication protocol, such as the ones discussed above, requires its own type of message processor at the home system and its own separate hardware system for implementation. Furthermore, as shown in FIG. 1, the communications provided and enabled by the home systems 12, 12A, 12B, 12C, and 12D are tracked to respective mobile systems 14, 14A, 14B, 14C, and 14D. However, a person who possesses mobile devices or units of different communication protocol types generally has subscriptions to the services for those types of communication, and these subscriptions and services require separate provisions of the services and usually separate management and billing. Presently, cross-over for processing and handling the various messages and responses for the

3

different communication protocols generally does not exist at this time. A system and method of unifying management of the various mobile communication protocols and of tracking a person with subscriptions to services of various mobile communication protocols do not exist. Furthermore, the unification of subscriptions of various communication services across various respective communication protocols under a single umbrella for each subscriber, user, or customer also does not exist at this time.

For example, if a person owns a cellular telephone and a mobile laptop computer with Internet access, then another person trying to reach that person has to separately dial the cellular telephone and separately send an e-mail message to communicate through the mobile laptop computer. A system and method of tracking the mobility communications of a person and allowing a unified manager to automatically try the various mobile communication protocols for which the person has subscription(s) does not exist. For example, such a method that does not exist involves a type of unified communication manager for tracking the communications mobility of the person by first trying to access the person's cellular telephone and then automatically sending the e-mail message to the person's mobile laptop computer if the person was not reached by cellular telephone.

Thus, each type of mobile communication protocol requires its own separate hardware system, management, and provision of service. Present tracking of mobile communications is generally to a mobile device or unit and is not linked in any way to a person. A person may own various mobile communication devices or units. In the present technology, the communications of all of the mobile systems are each maintained and managed separately and are not unified in any manner. A unified management of mobile communications would allow the tracking of mobile communications to a person. The concept of unification of communication mobility management irrespective of the various access technologies and protocols at the home system presently does not exist. Furthermore, the unification of subscriptions of various communication services across various respective communication protocols under a single umbrella for each subscriber, user, or customer also does not exist.

It would therefore be advantageous and desirable to provide a system and method for unifying the implementation and processing of mobile communications by various mobile devices and units and terminals that operate on various mobile communication protocols. It would also be advantageous and desirable to provide a unified mobility manager system and method that is used to provide such unified implementation and processing of mobile communications by various mobile devices and units and terminals that operate on various mobile communication protocols. It would still be advantageous and desirable to normalize messages from the various mobile devices and units and terminals, to process, handle, and respond to these normalized messages, and to convert the normalized responses back to respective communication protocols. It would also be advantageous and desirable to provide a unified hardware system for various types of mobile communication protocols. It would also be advantageous and desirable to provide a system and method that allow tracking of mobile communications to a person and to his/her subscription services for various mobile communication types. It would further be advantageous and desirable to unify maintenance and management of the mobile communications that are of various mobile communication protocol types. It would still further be advantageous and desirable to provide unification of

4

mobility management irrespective of the various access technologies and protocols that exist at the home system. It would still also be advantageous and desirable to unify subscriptions of various communication services across various respective communication protocols under a single umbrella for each subscriber, user, or customer.

#### SUMMARY OF THE INVENTION

It is therefore one object of the present invention to provide a system and method for unifying the implementation and processing of mobile communications by various mobile devices and units and terminals that operate on various mobile communication protocols.

It is also another object of the present invention to provide a unified mobility manager system and method that is used to provide such unified implementation and processing of mobile communications by various mobile devices and units and terminals that operate on various mobile communication protocols.

It is still a further object of the present invention to normalize messages from the various mobile devices and units and terminals, to process, handle, and respond to these normalized messages, and to convert the normalized responses back to respective communication protocols.

It is a further object of the present invention to provide a unified hardware system for various types of mobile communication protocols.

It is still another object of the present invention to provide a system and method that allow tracking of mobile communications to a person and to his/her subscription services for various mobile communication types.

It is still a further object to unify maintenance and management of the mobile communications that are of various mobile communication protocol types.

It is still another object of the present invention to provide unification of mobility management irrespective of the various access technologies and protocols that exist at the home system.

It is still also an object of the present invention to unify subscriptions of various communication services across various respective communication protocols under a single umbrella for each subscriber, user, or customer.

The foregoing objects are achieved as is now described. A system and method for unifying and handling network messages of various communication protocols from various mobile systems. The network messages are being handled at a home system for the various mobile systems. A gateway cluster has a number of gateways of different types of communication protocols. The gateway cluster receives the network messages of the respective communication protocols at the home system. The respective gateways convert the network messages to normalized messages by querying the categories, the data, and the network types of the normalized data for the mobile systems from which the network messages were respectively generated therefrom. A database system stores normalized data in categories. The normalized data at least includes data relating to the mobile systems and network types for the data. A unified mobility manager is coupled to and in communications with the gateway cluster and the database system. The unified mobility manager receives and processes the normalized messages, performs operations based on the normalized messages and on the categories, the data, and the network types of the normalized data, and formulates normalized responses responsive to the normalized messages. The nor-

5

malized responses are converted to network responses at the gateways and the network responses are sent to the respective mobile systems. A normalized data structure is provided for the normalized data, and the data structure generally comprises network message data, a category type for the data, and a network type that is reflective of the communication protocol from the network message.

The above as well as additional objects, features, and advantages of the present invention will become apparent in the following detailed written description.

#### BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself however, as well as a preferred mode of use, further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

FIG. 1 are prior art block diagrams of various individual and separate configurations of mobile communications wherein each of the diagrams shows an individual and separate hardware system that has a mobile system which uses a local serving system to locate and find its home system in order to enable its respective type of communication protocol and provide communications for the mobile system;

FIG. 2 is a block diagram showing an example topology of the present invention unified mobility manager (UMM) used to provide unified implementation and processing of mobile communications by various mobile systems that operate on various mobile communication protocols;

FIG. 3 is a chart of example categories of UMM information;

FIG. 4 is an example data organization chart for the authentication information category showing various normalized data defined for each network/personality type (i.e. each communication protocol);

FIG. 5 is a block diagram of a UMM/UDS message processor system that handles and processes UMM messages;

FIG. 6 is a flow chart of an algorithm for the handling and processing of UMM messages by the UMM/UDS message processor system and through a gateway cluster;

FIG. 7 is a chart of permanent and transient subscriber data used in the present invention; and

FIG. 8 is a chart of temporary transient subscriber data used in the present invention.

#### DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENT

For the present detailed description, the term "mobile" or "mobility" is used to define and describe a device, unit, terminal, or system that is able to be moved from one location to another location, whether the device, unit, terminal, or system is a wireless system that is movable or a wireline system that is movable. Furthermore, for this present detailed description, the phrase "mobile communication" or "mobility communication" is used to define and describe communication protocols that track movement of such "mobile" devices, units, terminals, or systems from one location to another location and that provide and support communications for such "mobile" devices, units, terminals, or systems (i.e. whether these devices, units, terminals, or systems are wireless systems that are movable or wireline

6

systems that are movable). The present invention described herein is not in any way limited to any particular communication device, unit, terminal, system, or respective communication protocol, and the present invention may be utilized with any device, unit, terminal, or system that is able to be moved from one location to another location (whether a wireless system or a wireline system) and with any communication protocol that tracks, provides, and supports communications for such devices, units, terminals, or systems that are movable from one location to another location.

The present invention is a system and method for unifying the implementation and processing of mobile communications by various mobile devices and units and terminals ("mobile systems") that operate on various mobile communication protocols. With reference now to the figures and in particular with reference to FIG. 2, a unified mobility manager (UMM) 30 is shown. The UMM 30 unifies implementation and processing of mobile communications by various mobile systems, such as cellular/mobile telephones 28, pagers 29A, personal computers (PCs) 29B, Personal Digital Assistants (PDAs) 29C, etc., that operate on different mobile communication protocols, such as North American Cellular (NAC) or ANSI41 18, Group Special Mobile (GSM) network 20, a data network 22, a Public Switched Telephone Network (PSTN), etc. The UMM 30 is also adapted to support any other future network or protocol 26 as shown in FIG. 2. FIG. 2 shows that the UMM 30 provides a unified hardware system 30 that implements and processes messages and provides responses for various types of mobile communication protocols. The UMM 30 and the system and method of the present invention allow tracking of mobile communications to a person and to his/her subscription services for various mobile communication types. Thus, maintenance and management of the mobile communications that are of various mobile communication protocol types are unified.

As shown in FIG. 2, the UMM 30 provides a mobility manager platform that is capable of receiving messages for different networks (such as NAC/ANSI41 18, Group Special Mobile (GSM) network, cable modem network, mobile Internet protocol (IP), Public Switched Telephone Network (PSTN), data network, IS-136 Time Division Multiple Access (TDMA), IS-54B Time Division Multiple Access (TDMA), Advanced Mobile Phone System (AMPS), Code Division Multiple Access (CDMA), and any other such present or future technologies that is able to track, provide, and support mobile communications of mobile devices, units, terminals, or systems that move from one location to another location and that provide voice and data applications or any other information exchange to a mobile customer or user). The UMM 30 provides a unified mechanism of dealing with mobility communications for users/customers. The UMM 30 provides a logical common point for all mobility related messages regardless of the access mechanism at the home system. The reuse of technology is maximized by providing an Unified Directory Services (UDS) system 31 (see FIG. 5). The UMM 30 and UDS 31 will be discussed in more detail later. The UMM 30 and UDS 31 allow the creation of a common addressing mechanism, low level message parsing, state machine, etc. The present invention is able to provide and use context specific handlers as needed to support a specific protocol. The UMM 30 receives control messages that arrive in various networks and provide mobility management operations such as addressing, location, tracking of the user and his/her device(s), routing, etc. The UMM 30 minimizes the need for separate hardware and software platforms and

7

creates a unified, consistent mechanism for services provided to the users for mobile systems of different types of communication protocols. The UMM 30 provides unification of mobility management irrespective of the various access technologies and protocols that exist at the home system.

FIG. 2 shows that the UMM 30 involves profile integration and provides a common point for all mobility related messages regardless of access technology or protocol. The profile integration is the combining of various communication protocols into a common representation or normalized data. The profile integration is accomplished by looking for common mobility management attributes in various protocols and creating an independent normalized protocol representation. The common point for all mobility related messages is provided by the UMM 30. The UMM 30 is able to handle the mobility related messages in two general ways: 1) It can receive and handle all control messages in their respective communication protocol formats (i.e. ANSI41, Mobile Application Part (MAP), Mobile Internet Protocol (MIP), Next Generation (NG), Authentication Authorization and Accounting (AAA), nomadic wired mobility, or any other such present or future technologies that is able to track, provide, and support mobile communications of mobile devices, units, terminals, or systems that move from one location to another location and that provide voice and data applications or any other information exchange to a mobile customer or user) and 2) It can convert and handle all control messages in their respective communication formats (i.e. ANSI41, Mobile Application Part (MAP), Mobile Internet Protocol (MIP), Next Generation (NG), Authentication Authorization and Accounting (AAA), nomadic wired mobility, or any other such present or future technologies that is able to track, provide, and support mobile communications of mobile devices, units, terminals, or systems that move from one location to another location and that provide voice and data applications or any other information exchange to a mobile customer or user) into a normalized protocol that encompasses all mobility related functions and operations. These two options result in two respective conversions: 1) Protocol conversions, where only essential elements required for basic mobility management of legacy functions are converted and 2) Mobility management conversion, where only essential components of legacy mobility management functions are ported to the new UMM 30.

The present invention requires the unification of the information that the UMM 30 needs to access to be in a normalized manner or fashion so that the UMM 30 is able to, in fact, access this data. Therefore, a unified data storage method is required and disclosed for the present invention. The information needed by the UMM 30 needs to be analyzed and categorized.

With reference now to the figures and in particular with reference to FIG. 3, a chart 29 shows some example categories 32 of information needed by the UMM 30. Some common categories 32 of UMM information for various access technologies or communication protocols include but are not limited to location tracking information 34, authentication information 36, service level agreement 38, equipment identity 40, personal information and surveillance information 42, authorization information 44, and terminal capabilities 46. The present invention is not in any way limited to the specific categories disclosed, and any suitable category may be used in conjunction with the present invention.

After these common categories 32 have been established, data 50 pertaining to each category for each network/

8

personality type 48 (to each category within each type of communication protocol) is established, maintained, and constantly updated. With reference now to the figures and in particular with reference to FIG. 4, an example data organization chart 47 for the authentication information category 36 is shown. FIG. 4 shows the chart 47 with various normalized data 50 defined for each network/personality type 48 for the authentication information category 36. The data 50 is normalized data which is readable and usable by the UMM 30. For example, the chart 47 shows that for the network/personality of ANSI41 18 (such as for a cellular telephone), the normalized data 50 for the authentication information 36 is defined as Authentication Algorithm Version (AAV), Authentication Capability (AuthCap), Authentication Key (Akey), and Signaling and Messaging Encryption (SME) (NAC) wherein this normalized data 50 is used by the UMM 30 in order to handle and process authentication information for an ANSI41 network/personality. The chart 47 further shows that for the cable network/personality (such as for cable modem), the normalized data 50 for the authentication information 36 is defined as Cable Modem specific authorization wherein this normalized data 50 is used by the UMM 30 in order to handle and process authentication information for a cable network/personality. The chart 47 also shows that for the network/personality of GSM 20 or GSM GPRS, the normalized data 50 for the authentication information 36 is defined as Authentication (Auth.) triplets, Random Number (RAND/SRES), and Authentication Key (Kc) wherein this normalized data 50 is used by the UMM 30 in order to handle and process authentication information for a GSM or GSM/GPRS network or personality. The chart 47 still further shows that for the mobile IP network/personality (such as for a mobile computer with IP access), the normalized data 50 for authentication information 36 is defined as Authentication Certificates (Auth. Certificates) and encryption key wherein this normalized data 50 is used by the UMM 30 in order to handle and process authentication information for a mobile IP network/personality. Other charts similar to chart 47 would exist for the other categories (i.e. location tracking, service level agreement, equipment identity, personal information and surveillance information, authorization information, terminal capabilities, etc.) for correlating normalized data for each category to respective networks/personalities. The present invention is not in any way limited to the specific data category, network/personality, organization or normalization of data disclosed, and any suitable category, network/personality, data organization or normalization of data is able to be used in conjunction with the present invention. Furthermore, any present or future communication protocol that requires mobility management and that tracks, supports, and provides communications of a mobile device, unit, terminal, or system, whether it is a wireless system or a wireline or wired system, is able to be used in conjunction with the present invention.

Thus, all information pertaining to the various networks/personalities are stored in the same or similar format as illustrated in the example of FIG. 4. The generic representation of the format (as illustrated in chart 47 of FIG. 4) is as follows: network/personality type followed by the data for that network/personality type. The identification and correlation of the network/personality type to the data allows conversion from the network specific messages to normalized data and vice versa as well. Any data that is already common to all networks is unlabelled and is able to be extracted and directly accessed for all network types.

With reference now to the figures and in particular with reference to FIG. 5, the Unified Mobility Manager/Unified

Directory Services (UMM/UDS) message processor system 72 that handles and processes UMM messages that were converted from network specific messages is shown. Referring to FIGS. 1 and 5, the UMM/UDS message processor system 72 is located in the home system 12 and replaces the home message processor 15 located therein. Referring to FIG. 5, the UMM/UDS message processor system 72 comprises three main sub-systems: the gateway cluster 74, the Unified Mobility Manager (UMM) 30, and the Unified Directory Services (UDS) 31. The gateway cluster 74 comprises a number of gateways for various communication protocols (i.e. networks/personalities 48). These gateways receive network specific messages from the serving system 10 (i.e. see block diagram 5 of FIG. 1). The UMM 30 manages, retrieves, and performs all operations requested by network specific messages. The UDS 31 allows respective operations to be performed on it by the UMM 30, and the UDS 31 supplies all of the information needed to the UMM 30 by supplying the Network Specific Identifier (NSI), the Network Transparent Identifier (NTI), and the specific network/personality type 48 along with any common information that the user needs.

The present invention involves the implementation of a UMM message processing and handling method, which is illustrated in FIG. 5. The first step of the method, which generally occurs at location 84, involves the network specific messages being received by the gateway cluster 74 and directed to respective types of gateways depending on the type of messages received. For example, if a specific ANSI41 or NAC message is received at the gateway cluster 74, then the message is forwarded, normalized, and processed by the ANSI41 or NAC gateway 76. If a specific GSM message is received at the gateway cluster 74, then the message is forwarded, normalized, and processed by the GSM gateway 78. If a specific Mobile Internet Protocol (MIP) message is received at the gateway cluster 74, then the message is forwarded, normalized, and processed by the MIP gateway 76. This step of processing messages is performed in the same manner for any other types of messages at respective types of gateways. The second step of the method, which occurs at location 86, involves converting the network specific identities of the specific message to a network transparent identifier (NTI). The respective gateway performs the conversion, and the conversion is accomplished by looking up a conversion table of information, such as a local cache or UDS lookup table (i.e. conversion table may be derived from category charts, such as the chart 36 in FIG. 4). The conversion is generally from a Network Specific Identity (NSI) to a Network Non-specific Identity (NNI) with generic operations requested such as add, update, search, delete entries, etc. The normalized message having the NTI is forwarded to the UMM 30. The third step of the method, which generally occurs at location 88, involves the UMM 30 processing the normalized message and then performing the appropriate operation on the UDS 31. The information that is needed from the UDS 31 is obtained by supplying the Network Specific Identifier (NSI) along with the Network Transparent Identifier (NTI) and the specific network/personality type 48.

The fourth step of the UMM message processing method, which generally occurs at location 90, involves the UMM 30 retrieving and performing all the operation(s) requested by the network specific message that was/were from the mobile system 14 of the user. The UDS 31 searches through the unified or normalized data to look for any information pertaining to the user using the NSI and network/personality type 48 along with any common information that the mobile

system 14 of the user needs. An example of how the UDS data 96 is organized in the UDS 31 is shown in chart 95. The UDS data 96 may include the identification of the user 97. For that user 97, the network specific user identification (ID) 98, the location information 100, and other such information may be identified. The fifth step of the method, which generally occurs at location 92, involves the UMM 30 sending an intended response in the normalized manner to the appropriate and respective gateway within the gateway cluster 74. The respective gateway then converts the normalized response back to the corresponding communication protocol (i.e. network/personality), which provides a network specific response. The sixth step of the method, which generally occurs at location 94, involves sending the converted network specific response back through the serving system 10 and to the mobile system 14 and user (i.e. see block diagram 5 of FIG. 1).

With reference now to the figures and in particular with reference to FIG. 6, a flow chart of an algorithm 52 for handling and processing of UMM messages by the UMM/UDS message processor system 72 and through a gateway cluster 74 and for accessing the normalized data in the UDS 31 is shown. The algorithm 52 starts at block 54. The algorithm 52 moves to block 56 where a network specific message (i.e. from the serving system 10 as shown in FIG. 1) is received at the UMM 30. The network specific message contains a network specific identity (NSI). The algorithm 52 moves to block 58 where the UMM 30 translates this message from a NSI to a network transparent identifier (NTI) by querying the UDS 31. At this step at block 58, a simple unique match is generally sought. The algorithm 52 moves to block 60. At block 60, the UMM 30 constructs a normalized query for retrieving data using the NTI specified and a network/personality type 48, which is derived from the NSI of the network specific message. The algorithm 52 moves to block 62 where the UDS 31 returns all of the information pertaining to the NTI in response to the network specific message. The algorithm 52 moves to block 64 where the information is parsed to extract the network/personality specific attributes or data. The algorithm 52 moves to block 66 where the extracted data which constructs the network specific response for a network/personality type 48 is sent from the UMM 31 to the gateway cluster 74 and to the appropriate and respective gateway. The algorithm 52 ends and stops at block 68.

The present invention provides unification of subscriptions of various communication services across various respective communication protocols under a single umbrella for each subscriber, user, or customer. A user may have mobile devices, units, or terminals in which respective subscriptions to services for these devices, units, or terminals have been obtained, and the mobile devices or units or terminals are of different types of communication protocols. The present invention, which includes the gateway cluster 74, the UMM 30, and the UDS 31 at the home system 12 that processes network specific messages, may be implemented and used to track the various mobile communications of a user instead of the mobile systems or mobile devices, units, or terminals themselves. The normalized data provides a way for tracking all of the mobile communication protocols and mobile systems utilized by the user since all of the data is able to be stored in an unified manner (i.e. in a unified storage system and handled by a unified data manager), that is, respectively UDS 31 and UMM 30. All of the information for a user and his/her mobile systems able to be stored as normalized data in the UDS 31.

With reference now to the figures and in particular with reference to FIG. 7, a chart of permanent and transient

11

subscriber data 102, which is an example of normalized data for the present invention, is shown. FIG. 7 shows an example organization and format for the different types of permanent and transient data 102. The permanent and transient subscriber data 102 includes but is not limited to the following: 1) Top level User Identity 104 (i.e. Person, UserNAI, etc.); 2) User Identity/Terminal Identity 106 (i.e. NAI, IMSI, MI, and IP addresses); 3) Equipment Identity 108 (i.e. Serial Number (GSM and NAC), PC Identity (MACAddress), Intel Chip ID); 4) Personal Information 110 (i.e. name, address, etc.); 5) Service Level Agreements (SLA) 112; 6) Terminal capabilities/network capabilities 114 (i.e. Access specific or technology specific (IP enabled or access enabled)); 7) Location Information 116 (macro location) (i.e. COA (IP address of subnet), MSC ID (NAC only), MSC Number, VLR, SGSN Number (GSM), LSF NAI (IPM specific)); 8) Authentication Information 118 (i.e. AAV, AuthCap Akey, SME (NAC), Authentication Certificates, IPSec, IKE, AAV, Encryption key, etc. (IP networks), Security Parameter Index (SPI), Authentication triplets, RAND/SRES, and Kc (GSM network)); 9) Authorization 120 (i.e. Authorization Period, Geographic Authorization (NAC), RSZI lists, Call barring restrictions based on location/roaming, subscription restriction, etc., IP enabled networks (such as firewall, lifetime, subnets enabled)); 10) Surveillance and Call Trace 122 (i.e. GSM, NAC, IP, etc.); 11) Services 124 (i.e. Quality of Service Profile (such as GPRS, MIP, 1xRTT MIP), Origination services (such as local, national, toll, etc.), Termination services (such as forwarding services, conferencing, data delivery, call/data screening), Roaming services (such as location based services and restrictions, etc.).

With reference now to the figures and in particular with reference to FIG. 8, a chart of temporary transient subscriber data, which is another example of normalized data, is shown. FIG. 8 shows an example organization and format for the different types of temporary transient data 126. The temporary subscriber data 126 includes but is not limited to the following: 1) Micro level mobility information of the mobile/terminal 128; 2) Temporary transient information 130, which needs to be updated frequently where the UDS 31 may not be used (i.e. Location Area ID, Control channel data, mobile not reachable flag (NAC), MS Purged for GPRS, SGSN area restricted flag, etc. for GSM, MIP/1xRTT such as co-located address flag, broadcast data grams, identification).

While the invention has been particularly shown and described with reference to a preferred embodiment, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.

What is claimed is:

1. A method of unifying and handling network messages of various communication protocols from various mobile systems wherein the network messages are being handled at a home system for the various mobile systems comprising the steps of:

storing normalized data which at least includes data relating to the mobile systems and network types of the mobile systems correlated to the data wherein the normalized data are stored in categories in a database system,

receiving the network messages at the home system,

converting the network messages to normalized messages by querying the categories, the data, and the network types of the normalized data for the mobile systems

12

from which the network messages were respectively generated therefrom,

processing the normalized messages and performing operations based on the normalized messages and based on the categories, the data, and the network types of the normalized data,

formulating normalized responses responsive to the normalized messages,

converting the normalized responses to network responses, and

sending the network responses to the respective mobile systems.

2. The method according to claim 1, wherein the step of storing normalized data further comprises the step of:

storing the normalized data in a unified directory services system.

3. The method according to claim 1, wherein the step of receiving the network messages further comprises the steps of:

receiving the network messages at a gateway cluster that has a number of gateways for different types of communication protocols,

determining the different types of communication protocols for the network messages, and

directing each of the network messages to a respective one of the gateways based on each of the determined different types of communication protocols for the each of the network messages.

4. The method according to claim 3, wherein the converting steps are performed at the gateways.

5. The method according to claim 1, wherein the network messages contain network specific identities and wherein the step of converting the network messages further comprises the step of:

converting the network specific identities of the network messages to network transparent identifiers.

6. The method according to claim 5, wherein the step of converting the network specific identities further comprises the steps of:

providing lookup tables in the database system for the categories, and

using the lookup tables to convert the network specific identities to the network transparent identifiers.

7. The method according to claim 1, wherein the step of processing the normalized messages further comprises the steps of:

forwarding the normalized messages to a unified mobility manager, and

using the unified mobility manager to perform appropriate operations on the database system.

8. The method according to claim 7, wherein the network messages contain network specific identities and wherein the step of converting the network messages further comprises the step of converting the network specific identities of the network messages to network transparent identifiers and wherein the step of using the unified mobility manager further comprises the step of:

obtaining information needed for performing the appropriate operations by supplying the network specific identities along with the network transparent identifiers and the network types to the database system.

9. The method according to claim 1, wherein the step of processing the normalized messages further comprises the steps of:

using a unified mobility manager for retrieving and performing the appropriate operations, and

13

having the database system search through the normalized data to look for information relating to the mobile systems.

10. The method according to claim 9, wherein the network messages contain network specific identities and wherein the step of converting the network messages further comprises the step of converting the network specific identities of the network messages to network transparent identifiers and wherein the step of having the database system search further comprises the step of:

using the network specific identities, the network types, and common information for finding the information relating to the mobile systems.

11. A system for unifying and handling network messages of various communication protocols from various mobile systems wherein the network messages are being handled at a home system for the various mobile systems comprising:

a gateway cluster having a number of gateways of different types of communication protocols for receiving the network messages of the respective communication protocols at the home system and for converting the network messages to normalized messages at the gateways by querying the categories, the data, and the network types of the normalized data for the mobile systems from which the network messages were respectively generated therefrom,

a database system for storing normalized data in categories wherein the normalized data at least includes data relating to the mobile systems and network types of the mobile systems correlated to the data,

a unified mobility manager coupled to and in communications with the gateway cluster and the database system wherein the unified mobility manager receives and processes the normalized messages, performs operations based on the normalized messages and based on the categories, the data, and the network types of the normalized data, and formulates normalized responses responsive to the normalized messages, and wherein the normalized responses are converted to network responses at the gateways and the network responses are sent to the respective mobile systems.

12. The system according to claim 11, wherein the database system is a unified directory services system.

13. The system according to claim 11, wherein the respective gateways determine the different types of communication protocols for the network messages and direct each of the network messages to a respective one of the gateways based on each of the determined different types of communication protocols for the each of the network messages.

14. The system according to claim 11, wherein the network messages contain network specific identities and the gateways convert the network specific identities of the network messages to network transparent identifiers.

15. The system according to claim 14, wherein the database system comprises lookup tables for the categories wherein the lookup tables are used by the gateways to

14

convert the network specific identities to the network transparent identifiers.

16. The system according to claim 11, wherein the unified mobility manager performs appropriate operations on the database system.

17. The system according to claim 16, wherein the network messages contain network specific identities and wherein the gateways convert the network specific identities of the network messages to network transparent identifiers and wherein the unified mobility manager obtains information needed for performing the appropriate operations by supplying the network specific identities along with the network transparent identifiers and the network types to the database system.

18. The system according to claim 11, wherein the unified mobility manager is used for retrieving and performing the appropriate operations and the database system searches through the normalized data to look for information relating to the mobile systems.

19. The system according to claim 18, wherein the network messages contain network specific identities and wherein the gateways convert the network specific identities of the network messages to network transparent identifiers and wherein the database system uses the network specific identities, the network types, and common information for finding the information relating to the mobile systems.

20. A normalized data structure used for unifying and handling one of a number of network messages based on one of a number of communication protocols from various mobile systems wherein the network messages are being handled at a home system for the various mobile systems, said normalized data structure comprising:

data storage including:

data from the one of the network messages,  
one of a number of data categories for identifying a category type of the data, said data categories including location tracking information, service level agreement, personal information and terminal capabilities, and

one of a number of network types that identifies and correlates the respective one of the communication protocols to the data.

21. The normalized data structure according to claim 20, wherein the data categories further include authentication information, equipment identity, surveillance information, and authorization information.

22. The normalized data structure according to claim 20, wherein the network types include at least two network types among: a NAC/ANSI41 network, a Group Special Mobile (GSM) network, a cable modem network, a mobile Internet protocol (IP), a public switched telephone network (PSTN), a data network, an IS-136 time division multiple access (TDMA), an IS-54B time division multiple access (TDMA), an advanced mobile phone system (AMPS), and a code division multiple access (CDMA).

\* \* \* \* \*



US006330589B1

(12) **United States Patent**  
Kennedy

(10) **Patent No.:** US 6,330,589 B1  
(45) **Date of Patent:** Dec. 11, 2001

(54) **SYSTEM AND METHOD FOR USING A CLIENT DATABASE TO MANAGE CONVERSATION THREADS GENERATED FROM EMAIL OR NEWS MESSAGES**

#### FOREIGN PATENT DOCUMENTS

2 327 516 \* 2/1999 (GB).

#### OTHER PUBLICATIONS

Crocker, D., RFC 822, "Standard for the Format of ARPA Internet Text Messages," Network Working Group, pp. 1-47, Aug. 1992.\*

Comer, D., et al., "Conversation-Based Mail," ACM Trans. on Computer Systems, vol. 4, No. 4, pp. 299-319, Nov. 1986.\*

(List continued on next page.)

Primary Examiner—Zarini Maung

Assistant Examiner—Andrew Caldwell

(74) Attorney, Agent, or Firm—Kilpatrick Stockton LLP

(57)

#### ABSTRACT

A system including a client database for managing conversation threads generated from messages communicated in a client-server architecture is disclosed. The client database efficiently manages messages and optimizes communication between the client and server. In a specific embodiment, the messages include email messages from a SMTP server and news messages from a NNTP server. The conversation threads are generated for use in a MAPI format-sensitive application. The client database maintains a central archive of message-related information to support conversation threading of current and future messages downloaded from the server to the client. The client database supports efficient management of conversations so that conversation roots and nested replies are presented sequentially. When a message refers to another, unreceived message, the system creates a placeholder for the unreceived message in the client database. Using a placeholder eliminates the need to rethread all conversations after every download. The database includes data fields corresponding to specific fields of a typical MAPI format. The database also includes data fields to assist in providing more efficient and timely operation of retrieving and threading conversations from a local message store, such as a MAPI store. News messages can also be converted from a news conversation threading structure to a MAPI format.

30 Claims, 14 Drawing Sheets

(75) Inventor: Kevin Alan Kennedy, Redmond, WA (US)  
(73) Assignee: Microsoft Corporation, Redmond, WA (US)  
(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: 09/084,729

(22) Filed: May 26, 1998

(51) Int. Cl.<sup>7</sup> ..... G06F 15/16; G06F 7/00

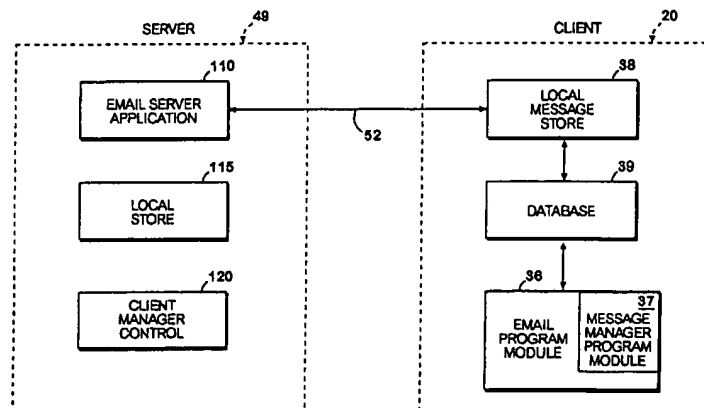
(52) U.S. Cl. .... 709/206; 707/104

(58) Field of Search ..... 709/203, 206;  
707/10, 102, 104; 379/93.24

(56) **References Cited**

#### U.S. PATENT DOCUMENTS

5,632,011	*	5/1997	Landfield et al.	709/206
5,757,669	*	5/1998	Christie et al.	709/205
5,793,972	*	8/1998	Shane	709/219
5,819,269	*	10/1998	Uomini et al.	707/7
5,826,022	*	10/1998	Nielsen	709/206
5,832,502	*	11/1998	Durham et al.	707/104
5,875,302	*	2/1999	Obhan	709/225
5,893,087	*	4/1999	Wlashin et al.	707/3
5,905,863	*	5/1999	Knowles et al.	709/206
5,909,679	*	6/1999	Hall	707/4
5,928,333	*	7/1999	Landfield et al.	709/245
5,937,162	*	8/1999	Funk et al.	709/206
6,020,884	*	2/2000	MacNaughton et al.	345/329
6,026,396	*	2/2000	Hall	707/4
6,073,137	*	6/2000	Brown et al.	707/104
6,092,101	*	7/2000	Birrell et al.	709/206
6,105,055	*	8/2000	Pizano et al.	709/204
6,134,582	*	10/2000	Kennedy	709/206
6,167,402	*	12/2000	Yeager	707/10



OTHER PUBLICATIONS

Shepherd, A., et al., "Strudel—An Extensible Electronic Conversion Toolkit," Proc. of the Conf. on Computer-Supported Cooperative Work, ACM, pp. 93–104, Oct. 1990.\*  
 Palme, J., et al., "SuperKOM—Design Considerations for a Distributed, Highly Structured Computer Conferencing System," Computer Communications, vol. 15, No. 8, pp. 509–518, Oct. 1992.\*  
 Hall, R., "INFOMOD: A Knowledge-Based Moderator for Electronic Mail Help Lists," Proc. of the 5th Int'l. Conf. on Information and Knowledge Management, ACM, pp. 107–114, Nov. 1996.\*  
 Levitt, Mark, "POP Goes the E-Mail," Copyright ©1996 International Data Corporation, published Sep. 1996, Document #12210, pp 1–9.  
 "What is IMAP!" article found on the World Wide Web at <http://www.imap.org/whatisIMAP.html>, The IMAP Connection, ©1996 The University of Washington, p. 1.

Gray, Terry, "Comparing Two Approaches to Remote Mailbox Access: IMAP vs. POP," article found on the World Wide Web at <http://www.imap.org/imap.vs.pop.brief.html>, Nov. 5, 1993, pp. 1–4.

Gray, Terry, "Message Access Paradigms and Protocols," article found on the World Wide Web at <http://www.imap.org/imap.vs.pop.html>, Aug. 28, 1995, pp. 1–10.

Crispin, M., "Internet Message Access Protocol—Version 4rev1," article found on the World Wide Web at <http://www.imap.org/docs/rfc2060.html>, Dec. 1996, p. 1.

Myers et al., "Post Office Protocol—Version 3," Carnegie Mellon, Dover Beach Consulting, Inc., Nov. 1994, pp. 1–8.

\* cited by examiner

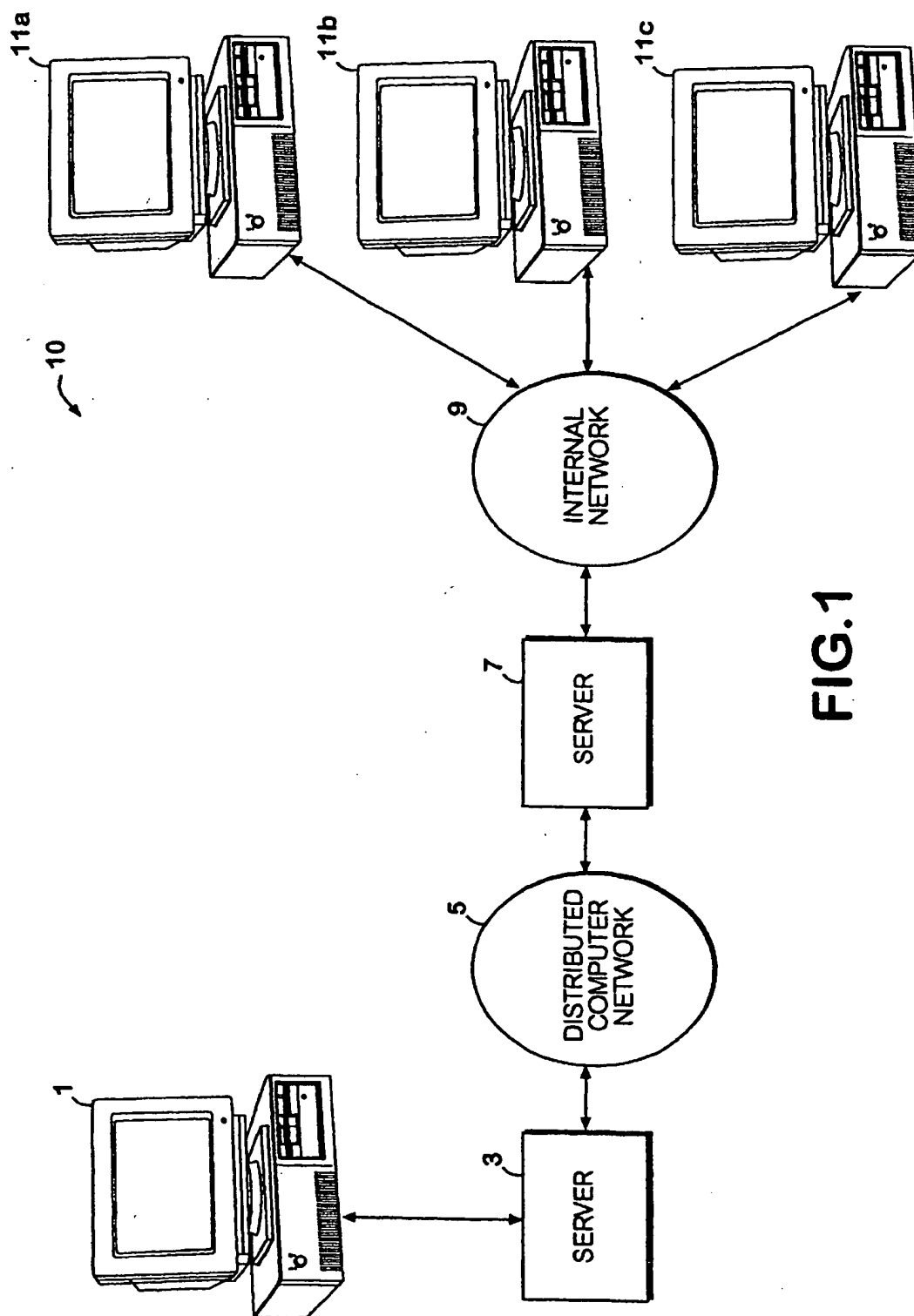
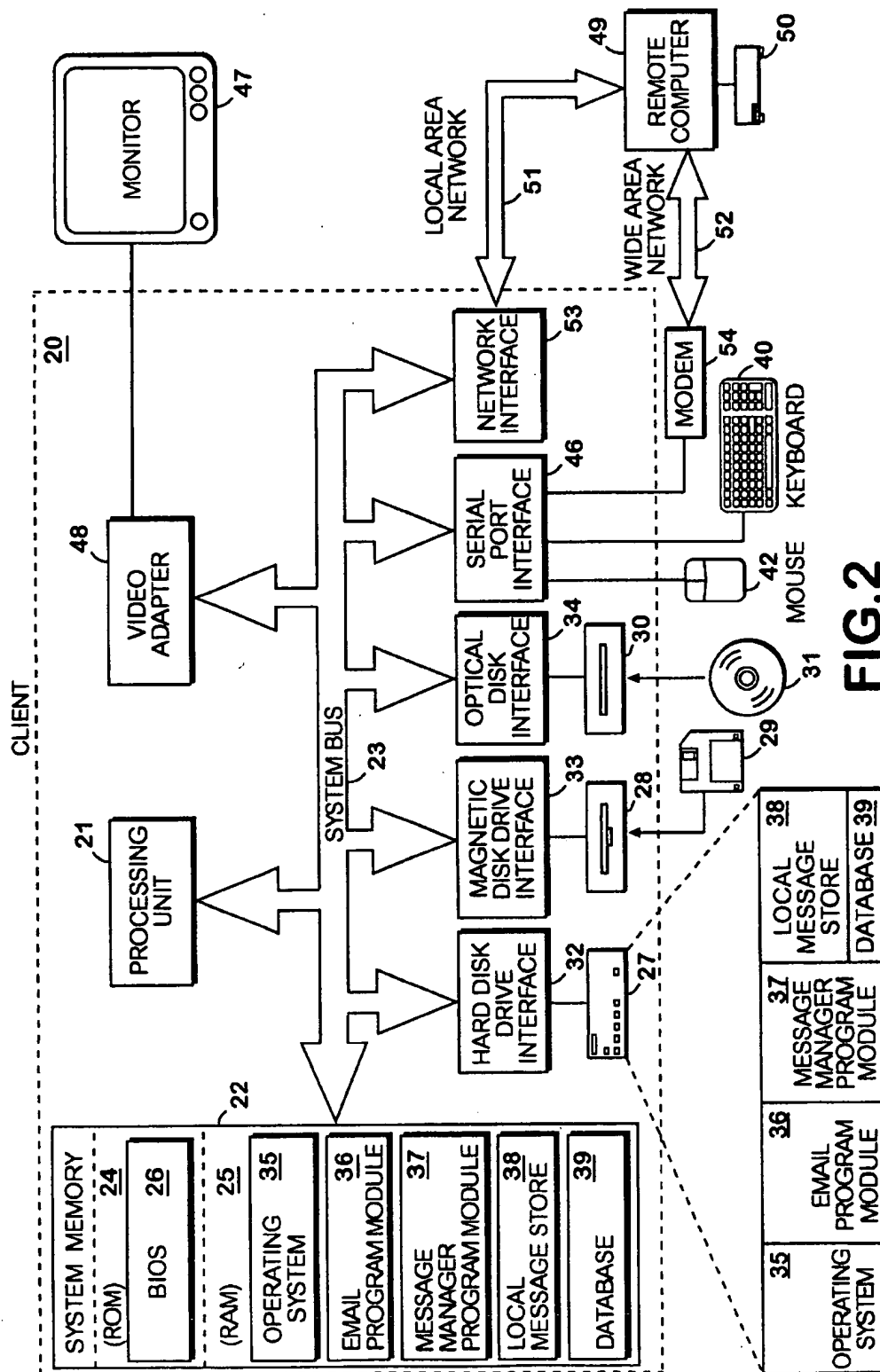


FIG.1



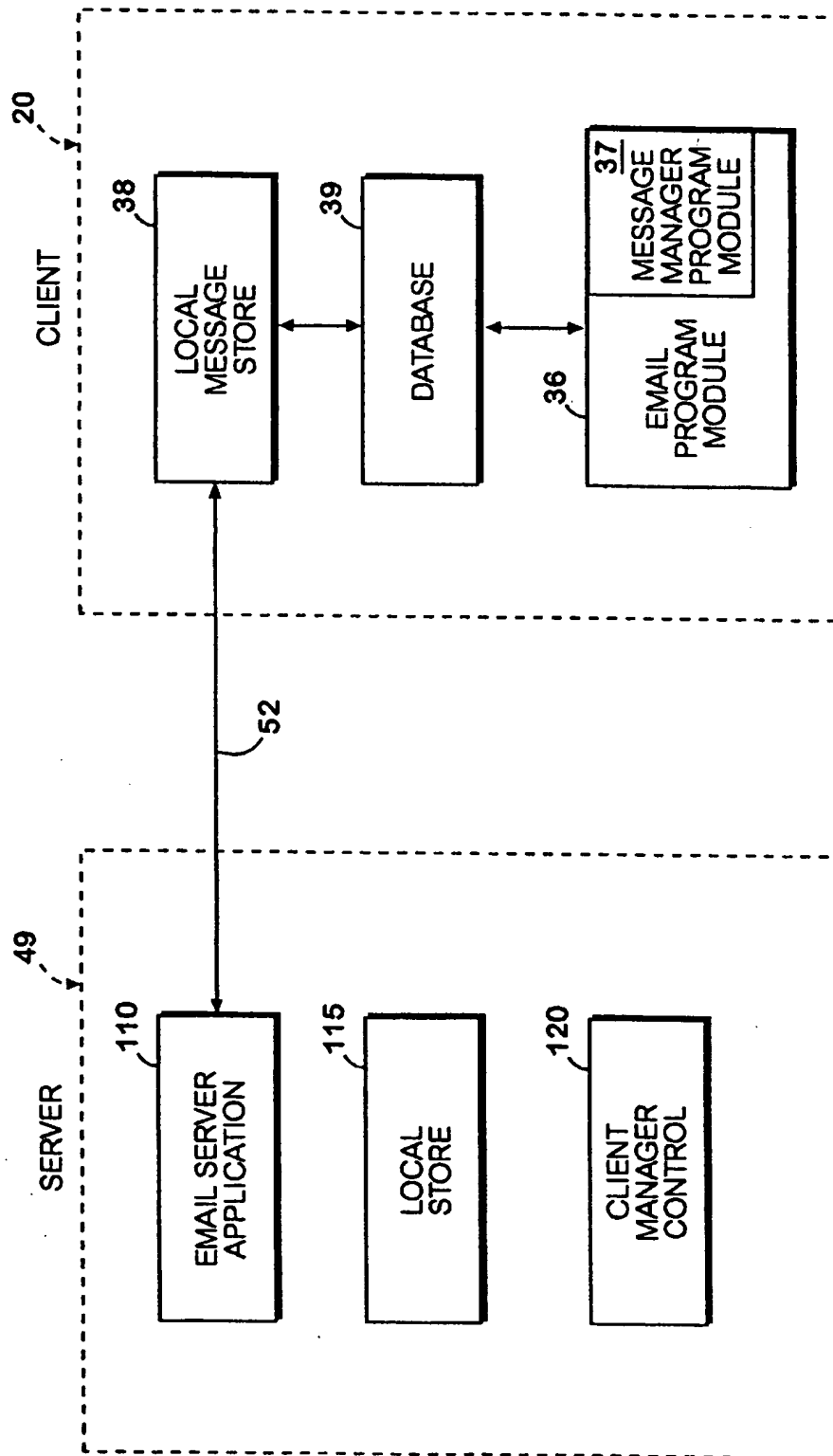


FIG. 3

## ARTICLE 100

<u>TITLE</u>	<u>MESSAGE ID</u>	<u>REFERENCES FIELD</u>
MESSAGE 1	A	- NO REFERENCE - 100
MESSAGE 2	B	< A > 102
MESSAGE 3	C	< A > < B > 104
MESSAGE 4	D	< A > 106

(SIMPLE CASE)

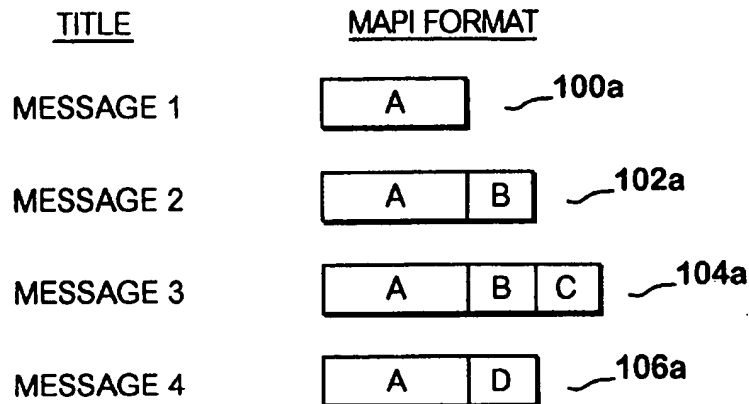
FIG.4a

## ARTICLE 200

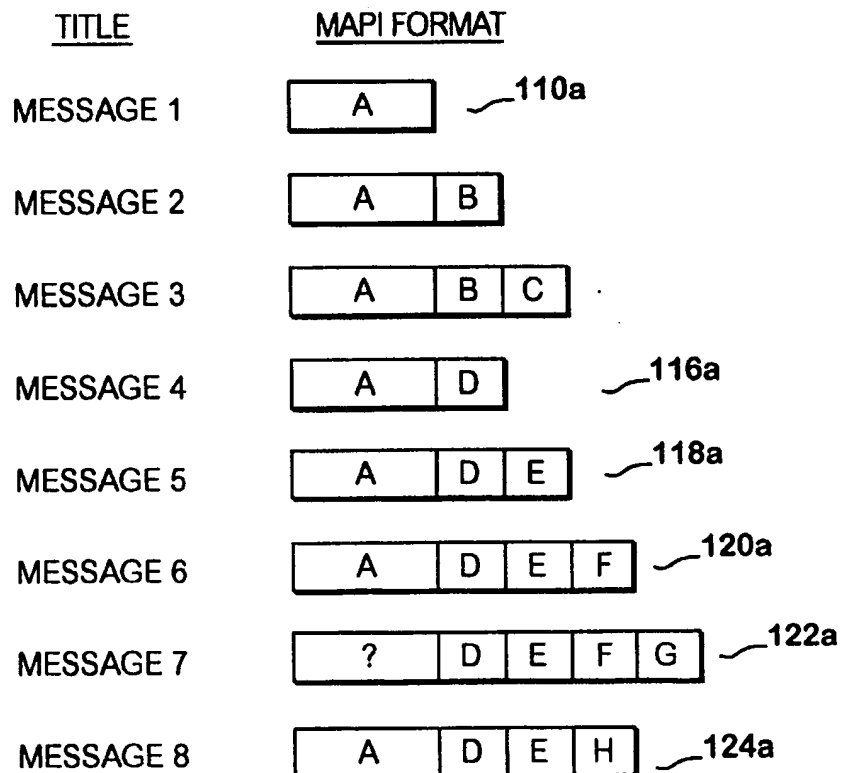
<u>TITLE</u>	<u>MESSAGE ID</u>	<u>REFERENCES FIELD</u>
MESSAGE 1	A	- NO REFERENCE - 110
MESSAGE 2	B	< A > 112
MESSAGE 3	C	< A > < B > 114
MESSAGE 4	D	< A > 116
MESSAGE 5	E	< A > < D > 118
MESSAGE 6	F	< A > < D > < E > 120
MESSAGE 7	G	< D > < E > < F > 122
MESSAGE 8	H	< A > < D > < E > 124

(COMPLEX CASE)

FIG.4b

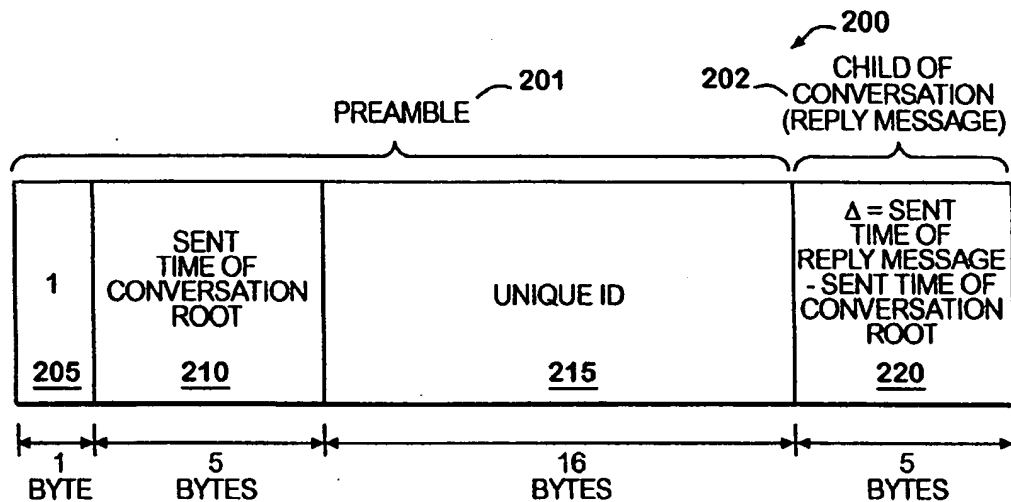


(SIMPLE CASE)

**FIG.4c**

(COMPLEX CASE)

**FIG.4d**



CONVERSATION INDEX

FIG. 5

FIG. 6a

Database 39

EID	ARTICLE NO.	TITLE	MESSAGE ID	PARENT ID	POSTED TIME	"ROOT" FLAG	"DIRTY" FLAG	"PLACE HOLDER" FLAG
1	100	MESSAGE 1	A	-	1/1/98 08:00:00	T	T	F
2	100	MESSAGE 2	B	A	1/1/98 08:01:00	F	T	F
3	100	MESSAGE 3	C	B	1/1/98 08:02:00	F	T	F
4	100	MESSAGE 4	D	A	1/1/98 08:03:00	F	T	F

FIG. 6a

FIG. 6b

Database 39

EID	ARTICLE NO.	TITLE	MESSAGE ID	PARENT ID	POSTED TIME	"ROOT" FLAG	"DIRTY" FLAG	"PLACE HOLDER" FLAG
1	100	MESSAGE 1	A	-	1/1/98 08:00:00	T	F	F
2	100	MESSAGE 2	B	A	1/1/98 08:01:00	F	F	F
3	100	MESSAGE 3	C	B	1/1/98 08:02:00	F	F	F
4	100	MESSAGE 4	D	A	1/1/98 08:03:00	F	F	F

FIG. 6b

DATABASE  
39

300 EID	305 ARTICLE NO.	310 TITLE	315 MESSAGE ID	320 PARENT ID	325 POSTED TIME	330 "ROOT" FLAG	335 "DIRTY" FLAG	340 "PLACE HOLDER" FLAG
1	200	MESSAGE 7	G	F	1/2/98 08:00:00	F	T	F
2	200	-	F	E	-	F	T	T
3	200	-	E	D	-	F	T	T
4	200	-	D	-	-	T	T	T

FIG.7a

DATABASE  
39

300 EID	305 ARTICLE NO.	310 TITLE	315 MESSAGE ID	320 PARENT ID	325 POSTED TIME	330 "ROOT" FLAG	335 "DIRTY" FLAG	340 "PLACE HOLDER" FLAG
1	200	MESSAGE 7	G	F	1/2/98 08:00:00	F	F	F
2	200	-	F	E	-	F	F	T
3	200	-	E	D	-	F	F	T
4	200	-	D	-	-	T	F	T

FIG.7b

DATABASE  
39

300 EID	305 ARTICLE NO.	310 TITLE	315 MESSAGE ID	320 PARENT ID	325 POSTED TIME	330 "ROOT" FLAG	335 "DIRTY" FLAG	340 "PLACE HOLDER" FLAG
1	200	MESSAGE 7	G	F	1/2/98 08:00:00	F	T	F
2	200	-	F	E	-	F	T	T
3	200	-	E	D	-	F	T	T
4	200	-	D	A	-	F	T	T
5	200	MESSAGE 8	H	E	1/3/98 08:00:00	F	T	F
6	200	-	A	-	-	T	T	T

FIG.7c

8/14

DATABASE  
39

300 EID	305 ARTICLE NO.	310 TITLE	315 MESSAGE ID	320 PARENT ID	325 POSTED TIME	330 "ROOT" FLAG	335 "DIRTY" FLAG	340 "PLACE HOLDER" FLAG
1	200	MESSAGE 7	G	F	1/2/98 08:00:00	F	F	F
2	200	-	F	E	-	F	F	T
3	200	-	E	D	-	F	F	T
4	200	-	D	A	-	F	F	T
5	200	MESSAGE 8	H	E	1/3/98 08:00:00	F	F	F
6	200	-	A	-	-	T	F	T

FIG.7d

DATABASE  
39

300 EID	305 ARTICLE NO.	310 TITLE	315 MESSAGE ID	320 PARENT ID	325 POSTED TIME	330 "ROOT" FLAG	335 "DIRTY" FLAG	340 "PLACE HOLDER" FLAG
1	200	MESSAGE 7	G	F	1/2/98 08:00:00	F	T	F
2	200	-	F	E	-	F	T	T
3	200	-	E	D	-	F	T	T
4	200	-	D	A	-	F	T	T
5	200	MESSAGE 8	H	E	1/3/98 08:00:00	F	T	F
6	200	MESSAGE 1	A	-	1/4/98 08:00:00	T	T	F

FIG.7e

DATABASE  
39

300 EID	305 ARTICLE NO.	310 TITLE	315 MESSAGE ID	320 PARENT ID	325 POSTED TIME	330 "ROOT" FLAG	335 "DIRTY" FLAG	340 "PLACE HOLDER" FLAG
1	200	MESSAGE 7	G	F	1/2/98 08:00:00	F	T	F
2	200	-	F	E	-	F	T	T
3	200	-	E	D	-	F	T	T
4	200	MESSAGE 4	D	A	1/4/98 08:01:00	F	T	F
5	200	MESSAGE 8	H	E	1/3/98 08:00:00	F	T	F
6	200	MESSAGE 1	A	-	1/4/98 08:00:00	T	T	F

FIG.7f

DATABASE  
39

300 EID	305 ARTICLE NO.	310 TITLE	315 MESSAGE ID	320 PARENT ID	325 POSTED TIME	330 "ROOT" FLAG	335 "DIRTY" FLAG	340 "PLACE HOLDER" FLAG
1	200	MESSAGE 7	G	F	1/2/98 08:00:00	F	T	F
2	200	-	F	E	-	F	T	T
3	200	MESSAGE 5	E	D	1/4/98 08:02:00	F	T	F
4	200	MESSAGE 4	D	A	1/4/98 08:01:00	F	T	F
5	200	MESSAGE 8	H	E	1/3/98 08:00:00	F	T	F
6	200	MESSAGE 1	A	-	1/4/98 08:00:00	T	T	F

FIG.7g

DATABASE  
39

300 EID	305 ARTICLE NO.	310 TITLE	315 MESSAGE ID	320 PARENT ID	325 POSTED TIME	330 "ROOT" FLAG	335 "DIRTY" FLAG	340 "PLACE HOLDER" FLAG
1	200	MESSAGE 7	G	F	1/2/98 08:00:00	F	F	F
2	200	-	F	E	-	F	F	T
3	200	MESSAGE 5	E	D	1/4/98 08:02:00	F	F	F
4	200	MESSAGE 4	D	A	1/4/98 08:01:00	F	F	F
5	200	MESSAGE 8	H	E	1/3/98 08:00:00	F	F	F
6	200	MESSAGE 1	A	-	1/4/98 08:00:00	T	F	F

FIG.7h

DATABASE  
39

300 EID	305 ARTICLE NO.	310 TITLE	315 MESSAGE ID	320 PARENT ID	325 POSTED TIME	330 "ROOT" FLAG	335 "DIRTY" FLAG	340 "PLACE HOLDER" FLAG
1	200	MESSAGE 7	G	F	1/2/98 08:00:00	F	T	F
2	200	MESSAGE 6	F	E	1/4/98 08:03:00	F	T	F
3	200	MESSAGE 5	E	D	1/4/98 08:02:00	F	F	F
4	200	MESSAGE 4	D	A	1/4/98 08:01:00	F	F	F
5	200	MESSAGE 8	H	E	1/3/98 08:00:00	F	F	F
6	200	MESSAGE 1	A	-	1/4/98 08:00:00	T	F	F

FIG.7i

DATABASE  
39

300 EID	305 ARTICLE NO.	310 TITLE	315 MESSAGE ID	320 PARENT ID	325 POSTED TIME	330 "ROOT" FLAG	335 "DIRTY" FLAG	340 "PLACE HOLDER" FLAG
1	200	MESSAGE 7	G	F	1/2/98 08:00:00	F	F	F
2	200	MESSAGE 6	F	E	1/4/98 08:03:00	F	F	F
3	200	MESSAGE 5	E	D	1/4/98 08:02:00	F	F	F
4	200	MESSAGE 4	D	A	1/4/98 08:01:00	F	F	F
5	200	MESSAGE 8	H	E	1/3/98 08:00:00	F	F	F
6	200	MESSAGE 1	A	-	1/4/98 08:00:00	T	F	F

FIG.7j

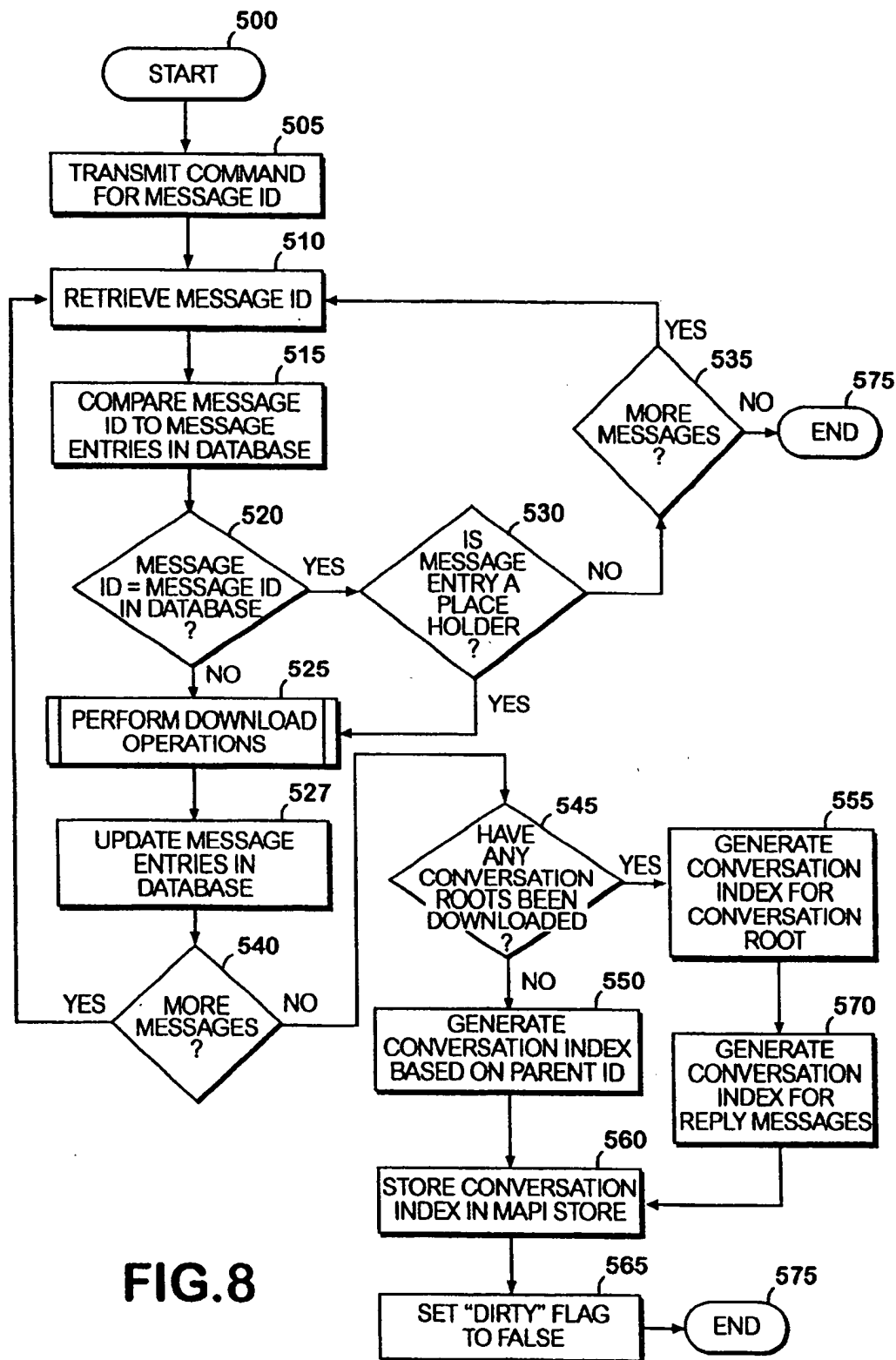


FIG. 8

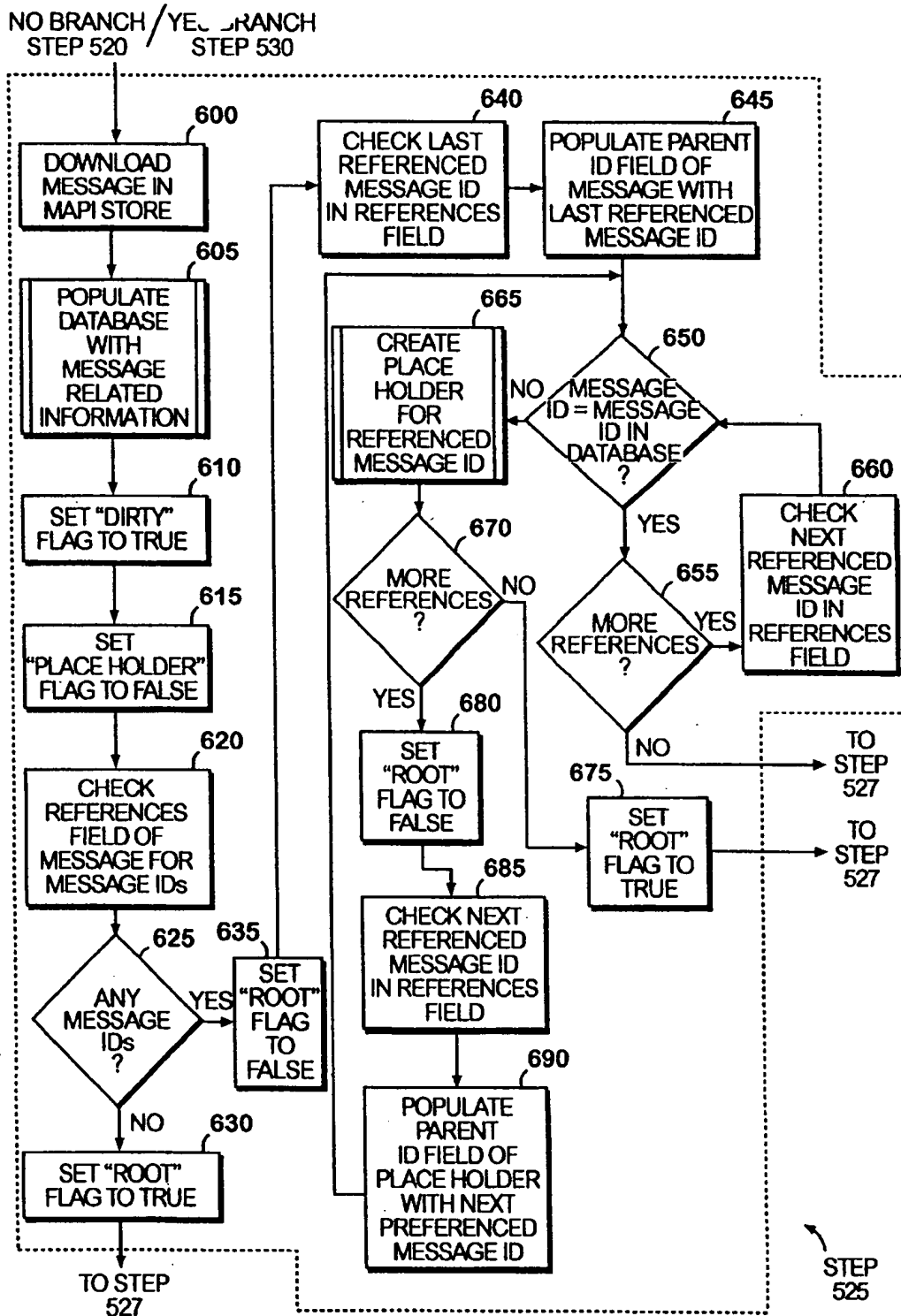
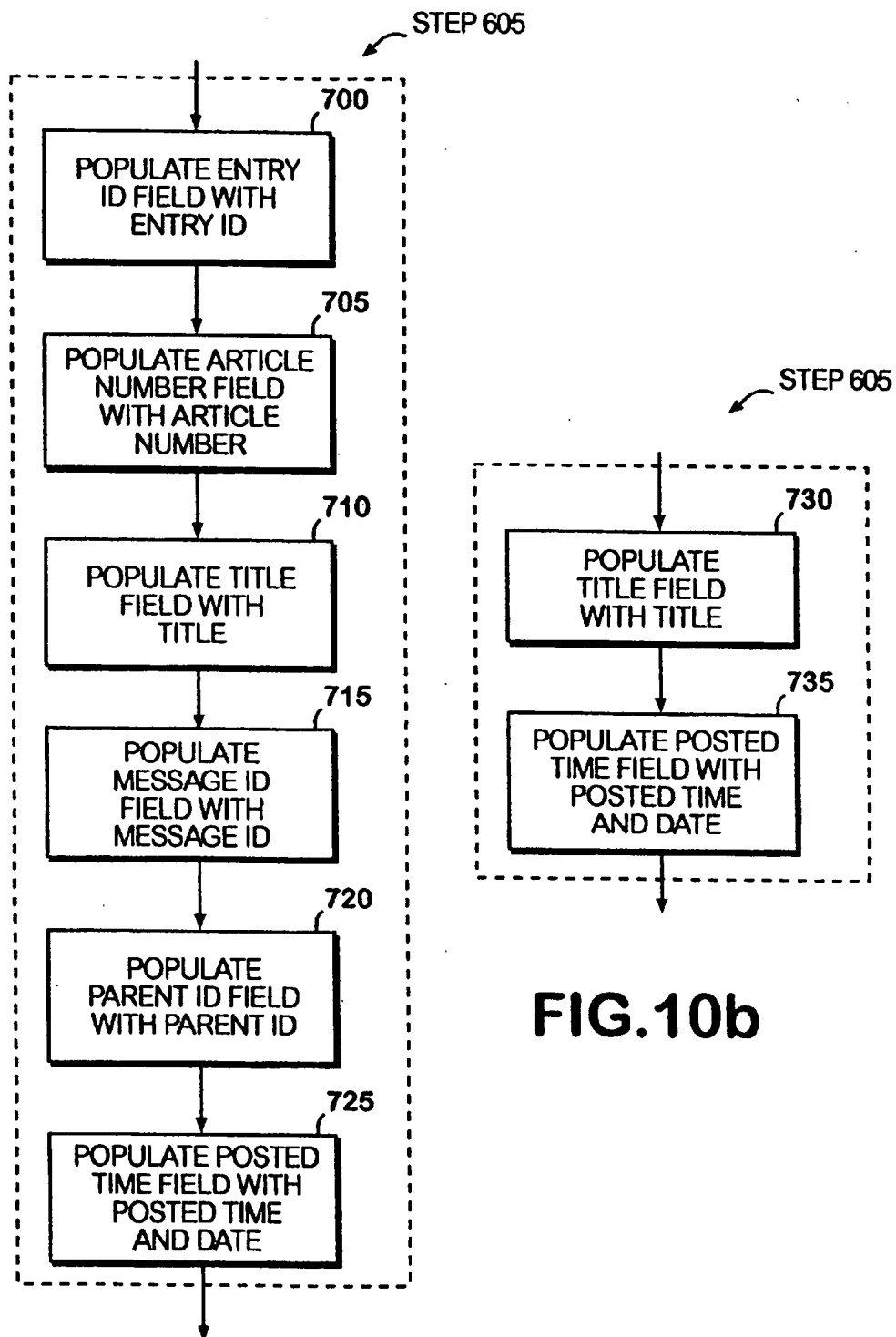


FIG. 9

## POPULATE DATA FIELDS IN DATABASE



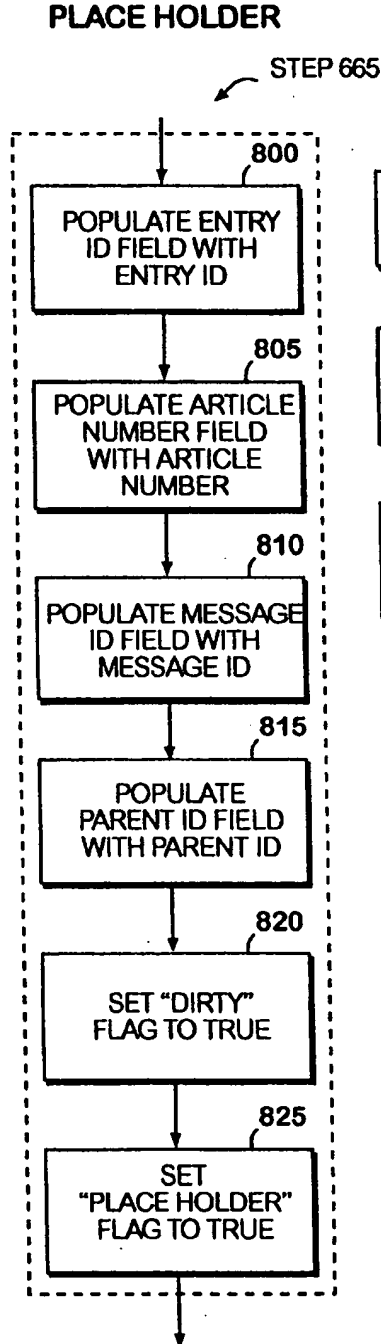
CREATION OF  
PLACE HOLDER

FIG.11

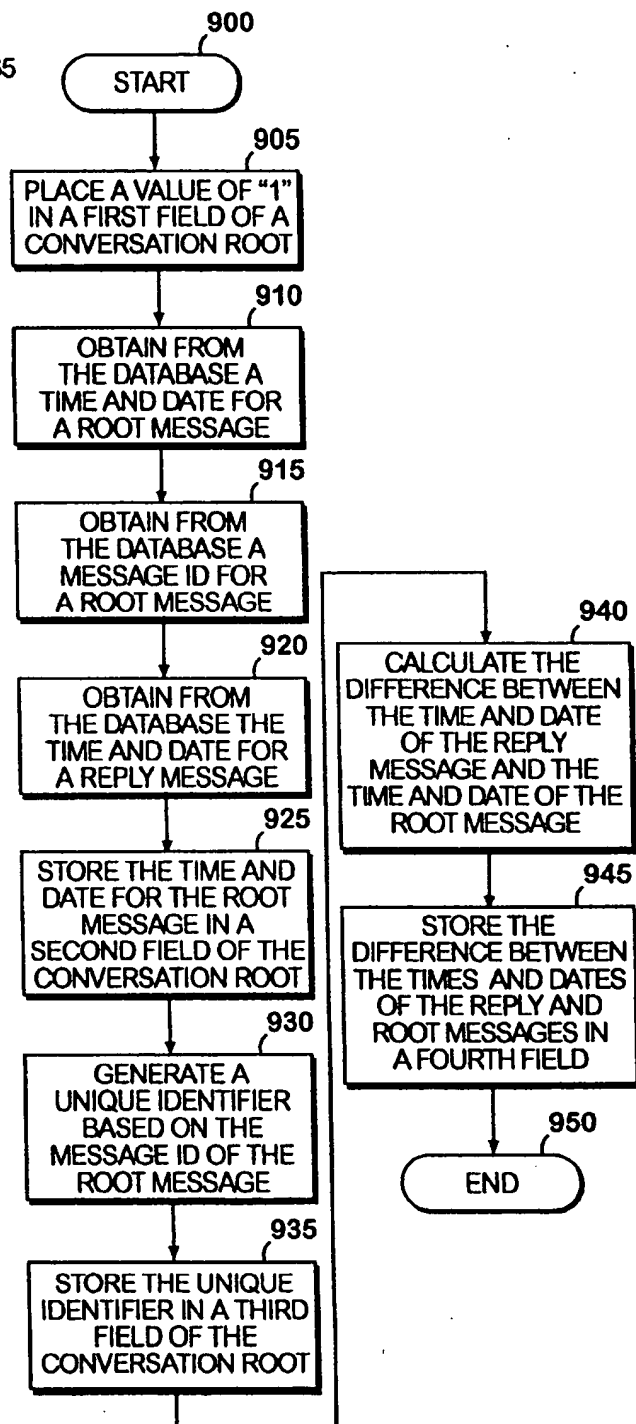


FIG.12

1

# SYSTEM AND METHOD FOR USING A CLIENT DATABASE TO MANAGE CONVERSATION THREADS GENERATED FROM EMAIL OR NEWS MESSAGES

## TECHNICAL FIELD

The present invention relates to communication and storage of electronic mail messages, and more particularly relates to managing conversation threads by using a client-based database.

## BACKGROUND OF THE INVENTION

Electronic mail (e-mail) is one of the most commonly used applications for distributed computer networks. The benefits of e-mail applications are numerous. Users can quickly communicate with one another. If a user is not available to pick up a message immediately, the message is stored until that user can review the stored message at a later time. E-mail messages also provide a quick and easy way to package information such as sales reports, graphics, and other data for transfer to another user by simply attaching the information to the message. These days, business users increasingly rely on e-mail messages to share ideas, transmit documents, schedule meetings, and perform a multitude of other everyday tasks.

These tasks may be accomplished by a variety of software programs. For example, e-mail programs facilitate the transmission of messages between users. Messaging-enabled scheduling programs allow users to request and schedule meetings and appointments via electronic messages. Computer programs known as desktop information managers attempt to coordinate the growing stream of electronic communications by incorporating e-mail, a calendar, task management, contact management, notes, and journal features into a single application program.

The increased reliance on electronic messaging has resulted in a great increase in the number of electronic messages a user sends and receives daily. These messages can include news messages, as well as general mail messages. Users who send and receive a large number of e-mail messages would like an effective way to process their e-mail without spending a lot of time sorting through their in-box, deleting, filing, forwarding, and responding to their messages. Hence, a major problem with e-mail is that a user can become inundated with messages without an efficient and effective means to manage them, especially with respect to news messages.

Typically, news messages are formatted in a news conversation threading structure or tree structure, which is a conversation threading scheme used by some electronic message systems. This tree structure generally consists of complex message references with correspondingly unique message identifiers. In this conversation threading scheme, related news messages are grouped together forming a conversation. A conversation will generally consist of a conversation root and nested replies or reply messages. These nested replies or reply messages are commonly referred to as children of the conversation root. Based on the message references and identifiers, the tree structure is arranged so that the conversation can retain its original order. Specifically, the tree structure is arranged so that the conversation root is displayed first and reply messages are displayed thereafter in the order in which each is received. The tree structure also accommodates reply messages that stem from other reply messages and so forth. This form of conversation threading allows messages to be retrieved from a server in a logical order.

2

Other systems have managed conversation threading by generating a conversation index. The conversation index consists of a preamble for uniquely identifying the conversation root and additional parameters for each child of the conversation root. In this conversation threading scheme, a group of related messages making up a conversation generally have the same preamble. The parameters for related reply messages, whether a reply message is linked to the conversation root or another reply message, are derived from the conversation root. In this way, the group of related messages are linked to each other and message order can be maintained.

Oftentimes in these prior systems, the integrity of the message order can be difficult to maintain if all related messages are not retrieved at one time during a download operation between a server and a client. In addition, these systems generally must rethread all conversations after every download operation. These systems typically do not have a means for tracking messages so that message order can be maintained and messages do not have to be rethreaded.

Another problem resulting from prior systems using different conversation threading schemes is converting one conversation threading scheme to another. For example, some systems use a MAPI format, where messages stored in a local message store, such as a MAPI store, are threaded by generating a conversation index. The MAPI format cannot handle news messages, which are typically formatted in a tree structure. Not only are these systems unable to accommodate a news message in a tree structure, they do not have a means for converting the news message to a compatible conversation threading scheme.

Therefore, there is a need for a system that optimizes communication with electronic mail servers. There is also a need for a single mechanism for managing messages on a user's local message store. There is a need for a system that is able to accommodate news messages. In addition, there is a need for a system that effectively and efficiently keeps track of messages that are downloaded from a server to a client. Also, there is a need for a system that is able to maintain the message order of a threaded conversation. There is an additional need for a system that is able to convert one conversation threading scheme to another.

## SUMMARY OF THE INVENTION

The present invention satisfies the above-described needs by providing a system and method for managing conversation threads communicated in a client-server architecture. Specifically, the present invention can generate conversation threading information for use in a MAPI format-sensitive application. Advantageously, the present invention efficiently manages messages and optimizes communication between a client and server by using a database, stored at the client. The database maintains a central archive of message-related information to support conversation threading of current and future messages downloaded from the server to the client. More particularly described, the present invention converts news conversation threading to a MAPI format for use in application programs which use a MAPI store.

The client-based database is used to support efficient management of conversations so that conversation roots (root messages) and nested replies (reply messages) are presented sequentially. Furthermore, the database includes data fields corresponding to specific fields of a typical MAPI format. The database also includes data fields to assist in providing more efficient and timely operation of retrieving

3

and threading conversations from a local message store, such as a MAPI store.

The present invention generally operates within the environment of a distributed computer system including a server and a client, where the client includes a local message store and a database. The present invention can manage conversation threads based on message-related information for a message having a message identifier and a references field, where the message-related information is stored in the database. During a client-server session, the message-related information corresponding to the message is retrieved from the server. Based on the message-related information, a determination is made as to whether the message has been previously downloaded from the server to the local message store located at the client. In response to determining that the message has not been previously downloaded from the server to the local message store, the message is downloaded from the server to the local message store, and data fields in the database are populated with the message-related information. A determination then is made as to whether the references field of the message is empty. In response to determining that the references field of the message is empty, indications are provided in the database that the message is a conversation root and that a conversation index has not been determined for the message. This entire process is performed for each remaining message on the server during this client-server session. Finally, the conversation index is generated for each downloaded message based on the conversation root.

The message-related information generally comprises the message identifier for identifying the message, an article number for identifying the number of the article for the message, a title for identifying the title of the message, and a parent identifier for identifying the parent of the message. The message-related information also can include a posted or sent time for indicating a time and date that the message is posted to a database for news messages, preferably housed in a news server, such as an NNTP server. The data fields for the client-based database typically comprise a message identification field for the message identifier, an article number field for the article number, a title field for storing the title of the message, a parent identification field for storing the parent identifier for the message, and a posted time field for storing the time and date of the message.

In another aspect, the present invention can generate a conversation index based on message-related information for a reply message. Preferably, the reply message is arranged in a news conversation threading structure and includes a message identifier and a reference, which includes a message identifier for a root message. The message-related information can be stored in a client-based database and includes the message identifier and a time and date for the root message and the reply message. Advantageously, the conversation index, which includes fields that are arranged in the MAPI format, supports conversation threading so that the root message and the reply message are grouped in a MAPI format for use in a MAPI format-sensitive application. To generate a conversation index from a reply message, a value, preferably "1", is placed in a first field of the conversation index. The database then is consulted to obtain the time and date for the root message, the message identifier for the root message, and the time and date for the reply message. The time and date of the root message is stored in a second field of the conversation index. A unique identifier is generated based on the message identifier of the root message and is stored in a third field of the conversation index. The difference between the time and date of the reply

4

message and the time and date of the root message is determined and stored in a fourth field of the conversation index. As a result, the conversation index represents the reply message in the MAPI format for use in the MAPI format-sensitive application.

Advantageously, the present invention optimizes communication with electronic mail servers by utilizing a client-based database. The present invention also accommodates news messages by converting the news messages to a conversation threading scheme that is compatible to MAPI format-sensitive applications. In addition, the present invention effectively and efficiently keeps track of messages that are downloaded from a server to a client and maintains the message order of a threaded conversation.

These and other objects, features, and advantages of the present invention may be more clearly understood and appreciated from a review of the following detailed description of the disclosed embodiments and by reference to the appended drawings and claims.

#### BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a client-server operating environment for an exemplary embodiment of the present invention.

FIG. 2 is a block diagram of a client system for an exemplary embodiment of the present invention.

FIG. 3 is a block diagram illustrating inter-operation of a client and server in accordance with an exemplary embodiment of the present invention.

FIGS. 4a and 4b are diagrams illustrating a conversation threading scheme for news messages known as a tree structure in accordance with an exemplary embodiment of the present invention.

FIGS. 4c and 4d are diagrams illustrating a generic conversion of the news messages in FIGS. 4a and 4b, respectively, to a MAPI formatted conversation index.

FIG. 5 is a diagram of a MAPI-formatted conversation index generated from message-related information maintained in a client-based database in accordance with an exemplary embodiment of the present invention.

FIGS. 6a and 6b are diagrams illustrating a message manager database for archiving messages for conversation threading operations in accordance with an exemplary embodiment of the present invention.

FIGS. 7a, 7b, 7c, 7d, 7e, 7f, 7g, 7h, 7i, and 7j, collectively referred to as FIGS. 7a-7j, are diagrams illustrating a message manager database for archiving messages for conversation threading operations in accordance with an exemplary embodiment of the present invention.

FIG. 8 is a flow diagram illustrating an exemplary method of operation utilizing a client-based database in accordance with an exemplary embodiment of the present invention.

FIG. 9 is a flow diagram illustrating an exemplary process of downloading a message from the server to the client.

FIGS. 10a and 10b, collectively referred to as FIG. 10, are flow diagrams illustrating an exemplary process of populating data fields within a database.

FIG. 11 is a flow diagram illustrating an exemplary process for creating a place holder in accordance with an exemplary embodiment of the present invention.

FIG. 12 is a flow diagram illustrating an exemplary process for generating a conversation index based on message-related information in a database.

#### DETAILED DESCRIPTION

Generally, the present invention provides a system for managing messages communicated within a client-server

5

architecture, such as a distributed computing environment represented by a client and a server. Specifically, the present invention provides a system for managing conversation threads communicated in the client-server architecture. In an exemplary embodiment, the invention is incorporated into the "MICROSOFT OUTLOOK '98" application program, which is produced and distributed by Microsoft Corporation of Redmond, Wash. The "MICROSOFT OUTLOOK '98" application program can manage e-mail, calendars, contacts, tasks and to-do lists, and documents or files on a hard drive. The present invention, preferably implemented as a message manager program module ("message manager"), uses a database, stored at the client, to maintain a central archive of message-related information to support conversation threading of current and future messages downloaded from the server to the client. The present invention also converts news messages in a news conversation threading structure to a MAPI format for use in application programs which use a MAPI store.

Referring now to the drawings, in which like numerals represent like elements throughout the several figures, aspects of the present invention and an exemplary operating environment will be described.

#### EXEMPLARY OPERATING ENVIRONMENT

The following discussion is intended to provide a general description of a suitable computing environment in which the invention may be implemented. While the invention will be described in the general context of an application program that runs on an operating system in conjunction with a personal computer and in connection with a server, those skilled in the art will recognize that the invention also may be implemented in combination with other program modules. Generally, program modules include routines, operating systems, application programs, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like.

The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices. Execution of the program modules may occur locally in a stand-alone manner or remotely in a client/server manner. Examples of such distributed computing environments include local area networks of an office, enterprise-wide computer networks, and the Internet.

The Internet, which is a global web of interconnected computers and computer networks, integrates local area networks (LANs) located in various entities, such as businesses, libraries, federal agencies, institutes of learning, and research organizations into a single communication network. The Internet uses a common communication protocol suite, known as a Transmission Control Protocol/Internet Protocol (TCP/IP), which was specifically designed for the interconnection of different computer systems. Internal and external networks are linked by routers that route data packets from a sending network to another router or a receiving network. Gateways handle data transfer and conversion of messages from a sending network to the protocols used by a receiving network. Typically, gateways refer to

6

devices that translate between applications. For example, e-mail gateways translate messages from one vendor's messaging program to another vendor's messaging program so that users with different e-mail programs can share messages over a network.

The Internet uses a message standard, known as a Simple Mail Transfer Protocol (SMTP), which works in conjunction with a user's e-mail program and defines the control messages used by two computers to exchange e-mail messages. Such controls include verification of proper connection, identification of sender, negotiation of transmission parameters, and message transmission. SMTP is responsible for 1) sending mail created by a local user to another computer and 2) receiving mail from other computers on the network and transferring it to the local user's e-mail program.

Typically, the computers connected to a wide area network such as the Internet are identified as either servers or clients. A server is a computer that stores files that are available to the other computers connected to the network. For example, an e-mail server manages message traffic and mail boxes for users, in addition to translation facilities or gateways that allow message exchange between different types of e-mail programs. A client is a computer connected to the network that accesses shared resources provided by a server. To obtain information from a server, a client makes a request for a file or information located on the server using a specified protocol. Upon reception of a properly formatted request, the server downloads the file or information to a local message store located at the client.

FIG. 1 illustrates a typical client-server environment 10 in which an exemplary embodiment of the present invention operates. A computer system or client 1, such as a conventional personal computer or any device operable to communicate over a network, is connected to an Internet server computer 3 ("server"). The server 3 is generally provided by an Internet service provider (ISP), which provides Internet access for a typical Internet user. The server 3 is connected to a distributed computer network 5, such as the Internet, and enables the client 1 to communicate via the distributed computer network 5.

The client 1 communicates via the combination of the server 3 and the distributed computer network 5 to a server 7, such as a communication or an e-mail server. In an exemplary embodiment, servers 3 and 7 support e-mail services, contain a message store for holding messages until delivery, and contain a translation facility or gateway for allowing users having different e-mail programs to exchange mail. The server 7 is connected to an internal network 9 and enables the client 1 to communicate with the clients 11a, 11b, and 11c via the internal network 9.

The clients 11a, 11b, and 11c are not only able to respond to a communication from the client 1, but are also able to initiate communication with the client 1. The clients 11a, 11b, and 11c can send information via the internal network 9 to the server 7. The server 7, in turn, forwards the information to the client 1 via the distributed computer network 5. The information is retrieved by the server 3 and can be forwarded to the client 1, when requested by the client 1.

With reference to FIG. 2, an exemplary system for implementing the invention includes a conventional personal computer 20, which serves as a client. The client 20 may represent any or all of the clients 1, 11a, 11b, and 11c illustrated in FIG. 1. The client 20 includes a processing unit 21, a system memory 22, and a system bus 23 that couples

7

the system memory to the processing unit 21. The system memory 22 includes read only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS), containing the basic routines that help to transfer information between elements within the client 20, such as during START-up, is stored in ROM 24. The client 20 further includes a hard disk drive 27, a magnetic disk drive 28, e.g., to read from or write to a removable disk 29, and an optical disk drive 30, e.g., for reading a CD-ROM disk 31 or to read from or write to other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide nonvolatile storage for the client 20. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD-ROM disk, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored in the drives and RAM 25, including an operating system 35, one or more application programs, such as an e-mail program module 36, other program modules, such as a message manager program module 37, a local message store 38, and a database 39 for supporting e-mail applications. A user may enter commands and information into the client 20 through a keyboard 40 and pointing device, such as a mouse 42. Other input devices (not shown) may include a pen, touch-operated device, microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers or printers.

The client 20 operates typically in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be an e-mail server (which includes one or more message stores), as described above in connection with FIG. 1, a file server (which includes one or more file stores), a router, a peer device or other common network node, and typically includes many or all of the elements described relative to the client 20, although only a memory storage device 50 has been illustrated in FIG. 2. The logical connections depicted in FIG. 2 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the client 20 is connected to the LAN 51 through a network interface 53. When used in a WAN networking environment, the client 20 typically includes a modem 54 or other means for establishing communications over the WAN 52, such as the Internet. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the client 20, or portions thereof, may be stored in the remote memory storage device. It will be appreciated

8

that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

With continuing reference to FIGS. 1 and 2, FIG. 3 is a diagram illustrating inter-operation of a client and server in accordance with an exemplary embodiment of the present invention. This exemplary embodiment is embodied in the "MICROSOFT OUTLOOK '98" application program, which is published by Microsoft Corporation of Redmond, Wash. The program helps users communicate through e-mail, phone support and group scheduling capabilities, and allows users to create and view information using a consistent interface. The "MICROSOFT OUTLOOK '98" program supports an exemplary program module, namely a message manager program module 37, for managing messages. The "MICROSOFT OUTLOOK '98" program also supports various Internet protocols, including, but not limited to protocols known in the art such as Internet Message Access Protocol (IMAP), version 3.0 of Post Office Protocol (POP3), Simple Mail Transfer Protocol (SMTP), Multipurpose Internet Mail Extensions (MIME), and Hyper Text Markup Language (HTML), and Network News Transfer Protocol (NNTP).

In FIG. 3, a remote computer 49 operates as a server and generally includes an e-mail server application 110, a local store 115, and a client manager control 120. In an exemplary embodiment, the remote computer 49, also described as a server, is a POP3 mail server, but it will be appreciated that the present invention is not limited to this type server. In the exemplary embodiment, the client 20 includes a local message store 38, a database 39, an e-mail program module 36, and a message manager program module 37 for facilitating message management and operation of the database 39.

With respect to the exemplary embodiment, the client 20 provides two modes of operation in connection with the server 49. These modes are a default mode and a "leave on server" mode. In the default mode, the client 20 sends a delete command to the server 49 to delete a message from the server 49 after the message has been downloaded to the client 20. In the "leave on server" mode, the client does not send a delete command to the server 49 after the message has been downloaded to the client 20, thereby allowing the message to remain on the server 49 although the message has been downloaded. The mode of operation generally is selected based on user-preference. Advantageously, an exemplary embodiment of the present invention optimizes the management of messages when the client 20 is in the "leave on server" mode, which is the preferred mode of operation for purposes of this discussion.

The server 49 houses e-mail messages from clients in the local store 115 while awaiting transmission to an appropriate destination. The e-mail server application 110 forwards messages over the WAN 52 from a sender client (not shown) to the client 20, upon request by the client 20. The client manager control 120 is a program used to set up computer systems, such as clients 1, 11a, 11b, 11c (FIG. 1), and 20 (FIG. 2) on the network. The client manager control 120 can also specify the addresses of the computer systems located on the network. In addition, the client manager control 120 typically facilitates the management of incoming and outgoing messages on the server. When a request for a message is made by the client 20 to the server 49, the e-mail server application 110 on the server 49 responds by retrieving the message from the local store 115 on the server 49 and by transmitting the message over the WAN 52 to the client 20. The message is then downloaded into the local message store 38 located at the client 20. The local message store 38 houses all downloaded messages from the server 49.

There are essentially two types of e-mail messages that are usually downloaded from the server 49 to the client 20. These message types are news messages and general mail messages. Although the present invention efficiently manages both types of messages, this embodiment of the present invention specifically addresses managing news messages.

News messages generally include items such as articles, bulletins, information, and data that are distributed to Internet users or subscribers through direct mailing on the Internet and through the USENET news system. The USENET news system provides a central repository of news messages in a database (not shown) and allows a subscriber to select those items he or she desires to read. The database (not shown) preferably is housed in a news server (not shown), such as an NNTP server. NNTP specifies the protocol for the distribution, inquiry, retrieval, and posting of news messages. NNTP is designed so that news messages can be stored at a central database at one host and can be accessed by subscribers from remote locations.

An application program, such as the "MICROSOFT OUTLOOK '98" program, provides a user-friendly interface for facilitating interaction with the database (not shown) for the news messages. The server 49 serves as an interface between the database for the news messages and the application program. Unfortunately, due to the conversation threading scheme for news messages, an application program that is MAPI format-based cannot handle downloading these news messages and maintaining their order. Consequently, an exemplary embodiment of the present invention provides the benefit of converting the conversation threading scheme of news messages to a conversation threading scheme that can be managed by a MAPI format-sensitive application.

This exemplary embodiment efficiently manages news messages and optimizes communication between a client and server by using a database, stored at the client. The database maintains a central archive of message-related information to support conversation threading of news messages downloaded from the server to the client. The database includes data fields corresponding to specific fields of a typical MAPI format. The database also includes data fields to assist in providing more efficient and timely operations of retrieving and threading conversations from a local message store, such as a MAPI store.

With continuing reference to FIGS. 1-3, FIGS. 4a and 4b illustrate a conversation threading scheme for news messages known as a tree structure. The tree structure is a common format for news messages. For purposes of this discussion, news messages are referred to herein as simply "messages". FIGS. 4a and 4b are used as examples of a simple case and a complex case, respectively, and are referred to throughout this discussion to demonstrate operation of an exemplary embodiment of the present invention in connection with FIGS. 6a and 6b and FIGS. 7a-7j.

Each message usually includes a message identifier or ID and a references field. The message ID is a identification string, which typically is derived from a header of the message itself, that uniquely identifies the message. For purposes of this discussion, a message ID is represented as a single character for simplicity. The references field identifies other correspondence which the message references. Specifically, the references field lists message IDs of any messages prompting the submission of the message. The purpose of the references field is to allow messages to be grouped into conversations by a program module, such as an application program or a browser. A message that does not

have any references in its references field is a conversation root or root message. As used herein, the terms "conversation root" and "root message" are interchangeable and represent an original message having no references. A message that has references in its references field may be referred to as a reply message. The message may also include a message title, an article number, and other items that are described in connection with FIGS. 5-12.

In FIG. 4a, a simple tree structure is illustrated for a group of messages associated with article number 100. In row 100, Message 1 has a message ID "A" and no references in its references field. Moreover, Message 1 is a conversation root. In row 102, Message 2 has a message ID "B" and a references field consisting of <A>. In row 104, Message 3 has a message ID "C" and a references field consisting of <A><B>. In row 106, Message 4 has a message ID "D" and a references field consisting of <A>.

Assuming that each message ID represents a message, the messages are typically linked in a tree-like format based on the contents of the references fields, as illustrated in FIG. 4a. Specifically, Message 1 is directly linked to Message 2 and Message 4, and Message 2 is directly linked to Message 3.

In FIG. 4b, a more complex tree structure is illustrated for a group of messages associated with article number 200. In row 110, Message 1 has a message ID "A" and no references in its references field. Moreover, Message 1 is a conversation root. In row 112, Message 2 has a message ID "B" and a references field consisting of <A>. In row 114, Message 3 has a message ID "C" and a references field consisting of <A><B>. In row 116, Message 4 has a message ID "D" and a references field consisting of <A>. In row 118, Message 5 has a message ID "E" and a references field consisting of <A><D>. In row 120, Message 6 has a message ID "F" and a references field consisting of <A><D><E>. In row 122, Message 7 has a message ID "G" and a references field consisting of <D><E><F>. In row 124, Message 8 has a message ID "H" and a references field consisting of <A><D><E>.

As illustrated in FIG. 4b, Message 1 is directly linked to Message 2 and 4; Message 2 is directly linked to Message 3; Message 4 is directly linked to Message 5; Message 5 is directly linked to Message 6 and Message 8; and Message 6 is directly linked to Message 7.

Advantageously, the exemplary embodiment converts the tree structures illustrated in FIGS. 4a and 4b into a MAPI format conversation index shown in FIGS. 4c and 4d, where FIG. 4c corresponds to FIG. 4a and where FIG. 4d corresponds to FIG. 4b. FIGS. 4c and 4d, used as examples of a simple case and a complex case, respectively, are referred to throughout this discussion to demonstrate operation of the exemplary embodiment in connection with FIGS. 6a and 6b and FIGS. 7a-7j.

With continuing reference to FIGS. 4c and 4d and turning to FIG. 5, a diagram illustrating a conversation index in accordance with an exemplary embodiment of the present invention is described. A conversation index 200, also known as a PR\_Conversation\_Index, is a conversation threading format used in a MAPI store. If the message is a reply message, the conversation index 200 typically includes a preamble 201 and a child of conversation 202. However, if the message is a conversation root, the conversation index 200 only includes the preamble 201. Once a preamble is generated for a conversation root, a conversation index can be generated for all reply messages having the same conversation root. The preamble of the conversation index 200 is common to all reply messages having the same conver-

11

sation root, as shown in FIGS. 4c and 4d. Advantageously, an exemplary embodiment of the present invention stores message-related information for generating a conversation index in the client-based database 39.

The conversation index 200 contains fields 205, 210, 215, and 220. In MAPI format, the size of the fields are as follows: the first field 205 is one byte; the second field 210 is five bytes; the third field is sixteen bytes; and a fourth field or reply message field 220 is five bytes. It is understood by those skilled in the art that a conversation index can have more than one reply message field, as illustrated in FIGS. 4c and 4d.

Fields 205, 210, and 215 make up the preamble. Preferably, a value of "1" is placed in the first field 205. The second field 210 is populated with the sent time of the conversation root. The sent time is the time and date that a message is sent to a database for news messages, preferably housed in a news server. The third field 215 is populated with a unique identifier or ID for the conversation root. The unique ID is generated from the message ID of the root message using a "hashing" process, which converts the arbitrary length message ID to a fixed length unique ID.

Finally, if a message is a reply message, the message has at least one reply message field 220. The reply message field 220 is populated with a difference "delta" equal to the difference between the sent time of the reply message and the sent time of the conversation root.

Each child of conversation 202 represents the reply message and each reference, except for the reference for the root message, in the references field of the reply message. For example, turning briefly to FIG. 4d, the conversation index 124a for Message 8 has a preamble derived from a root message (message ID "A") and three children of the conversation derived from each reference, message ID "D" and message ID "E", as well as the reply message itself, message ID "H".

For a place holder that is a conversation root, a preamble is generated by using a message ID from the references field of a different message. In addition, the sent time for a place holder is an arbitrary date and time in the past. Once the conversation index for a place holder conversation root is generated, a conversation index for a related message can also be generated. Place holders will be described in greater detail in connection with FIGS. 6-11.

In view of the foregoing, the present invention provides the benefit of threading conversations without complex referencing schemes like a tree structure. Furthermore, the present invention provides an efficient process for converting a tree structure to a MAPI format for use in a MAPI store. Moreover, by converting a message to the MAPI format, missing messages do not ruin the message order, as can occur in systems that use a tree structure. Additional features and advantages of the present invention are described in connection with the download operation and use of the database for storing message-related information for managing conversation threads.

During the download operation, data fields are populated within the database 39 with message-related information for conversation threading in association with the downloaded message. The information includes an article number for identifying the article, a message identifier for uniquely identifying the message, a message title for identifying the title of the message, and other message-related information that will be described in greater detail herein below with respect to FIGS. 6-11. The e-mail program module 36 provides facilities for creating, addressing, sending,

12

receiving, and forwarding messages, while the message manager program module 37 manages messages during download and conversation threading operations utilizing the database 39.

In prior systems, messages are typically rethreaded each time messages are downloaded. An exemplary embodiment of the present invention solves this problem by utilizing the database to keep track of downloaded messages and conversation threading information. The use of the database 39 is described in greater detail in connection with FIGS. 6a and 6b, collectively described as FIG. 6, and FIGS. 7a-7j, collectively described as FIG. 7.

With continuing reference to FIGS. 1-5, FIGS. 6a and 6b and FIGS. 7a-7j illustrate a client-based database for archiving messages relating to conversation threading in accordance with an exemplary embodiment of the present invention.

The database 39 can include multiple data fields, organized within an array structure, for maintaining message-related information. To support download and conversation threading operations, typical data fields of the database include: an entry identification (EID) 300, an article number 305, a title 310, a message identification (message ID) 315, a parent identification (parent ID) 320, a posted time 325, a "root" flag 330, a "dirty" flag 335, and a "place holder" flag 340.

Operation of the database 39 in connection with the message manager program module 37 (FIG. 3) is presented by way of two representative examples. The first example is a simple case in which a root message and reply messages are downloaded during a single client-server session. This example is illustrated in FIGS. 6a and 6b with continuing reference to FIGS. 3, 4a, and 4c. In this example, a connection is made between a client 20 (FIG. 3) and server 49 (FIG. 3). A user desires to check her download news messages and is typically prompted by the e-mail program module 36 (FIG. 3) to enter a password for access to the messages.

The server 49 contains four messages in its local store 115. These messages are entitled Message 1, Message 2, Message 3, and Message 4, as shown in FIG. 4a. Message 1 has no reference in its references field and has a message ID "A". Message 2 has a references field consisting of <A> and has a message ID "B". Message 3 has a references field consisting of <A><B> and has a message ID "C". Message 4 has a references field <A> and has a message ID "D".

Referring to FIG. 6a, the client 20 transmits a command to retrieve a message ID for a message. The server responds by transmitting to the client a message ID "A". As previously mentioned in connection with FIG. 4a, the message having a message ID "A" is a conversation root and, consequently, the message has no reference in its references field. The client checks the database 39 to determine whether there are any message entries having the message ID "A". In this example, there are no message entries having this message ID. As a result, the associated message is downloaded in the local message store 38, which in this case is a MAPI store.

During download operations, data fields 300, 305, 310, 315, 320, 325, 330, 335, and 340 within the database 39 are populated with message-related information associated with the message. Specifically, an entry identifier is populated in an EID field 300, an article number is populated in an article number field 305, a title is populated in a title field 310, the message ID is populated in a message ID field 315, a parent

13

ID is populated in a parent ID field 320, and a time and date or sent time are populated in a posted time field 325. In this representative example, the message-related information for a message entry associated with the message having a message ID of "A" includes: EID "1", article number "100", title "Message 1", message ID "A", and posted time "Jan. 1, 1998 08:00:00". As previously mentioned, the message is a conversation root. Therefore, the message does not have a parent ID, as shown in FIG. 6a, and a "root" flag is set to a true state in the "root" flag field 330.

The "root" flag exists in two states, namely a true state and a false state. The true state serves as an indicator that the message is a conversation root. The false state serves as an indicator that the message is not a conversation root. The message manager 37 keeps track of which messages are root messages for generating conversation indices. Once a conversation index is generated for a root message, reply messages that depend from the root message can be organized easily and efficiently.

In response to downloading the message to the local message store 38, a "dirty" flag is set to a true state in the "dirty" flag field 335. The "dirty" flag also exists in a true state and a false state. The true state of the "dirty" flag is indicative of a message that does not have a conversation index set in the local message store. The true state of the "dirty" flag is also indicative of the message being "dirtied" because a conversation index of another message from which the message depends has not been set or has become "dirtied". The false state of the "dirty" flag serves as an indicator that a conversation index has been generated for the message.

Further in response to downloading the message, a "place holder" flag is set to a false state in the "place holder" flag field 340. A place holder also exists in a true state and a false state. The true state serves as an indicator that the message is a place holder. The false state serves as an indicator that the message is not a place holder. Specifically, a place holder is a message entry for a message that has not yet been downloaded, which is referred to herein as a "non-downloaded message". The message manager is aware of a message's existence if a message ID for the non-downloaded message is referenced in a references field of an already downloaded message. Consequently, the message manager sets up a place holder to reserve a place for the non-downloaded message in the database. Any information about the non-downloaded message that can be gleaned from the already downloaded message is stored in appropriate data fields in the database as an incomplete message entry. If or when the non-downloaded message is downloaded to the client 20, the remaining data fields for the incomplete message entry are populated with the appropriate information. In this example, the place holder is set to a false state because the message has been downloaded and the message entry for the message is complete.

Referring to FIG. 6a, once Message 1 has been downloaded and the appropriate data fields have been populated, the message manager 37 determines whether there are any more messages to be downloaded from the server 49 to the client 20. In this example, there are three messages remaining on the server 49. The next message ID is retrieved and the downloading process is repeated, where a determination is made as to whether the message ID matches any message IDs in the database 39. The entire process is repeated for each of the remaining messages on the server.

In this example, Message 2 is downloaded and the data fields 300, 305, 310, 315, 320, 325, 330, 335, and 340 within

14

the database 39 are populated with message-related information associated with the message. The message-related information for a message entry associated with the message having a message ID of "B" includes: EID "2", article number "100", title "Message 2", message ID "B", parent ID "A", and posted time "Jan. 1, 1998 08:01:00". The "root" flag is set to a false state in the "root" flag field 330 because Message 2 is not a conversation root and depends from Message 1, as shown in FIG. 4a. Also, a "dirty" flag for Message 2 is set to a true state in the "dirty" flag field 335. In addition, a "place holder" flag is set to a false state in the "place holder" flag field 340.

It will be appreciated by those skilled in the art that in a tree structure format, the parent ID is typically the last referenced message ID in the references field of the message. In addition, the parent ID is preferably an internal pointer for pointing to the last referenced message ID. However, for purposes of this discussion, a single character is used to represent the parent ID for simplicity.

Next, Message 3 is downloaded and the data fields 300, 305, 310, 315, 320, 325, 330, 335, and 340 within the database 39 are populated with message-related information associated with the message. The message-related information for a message entry associated with the message having a message ID of "C" includes: EID "3", article number "100", title "Message 3", message ID "C", parent ID "B", and posted time "Jan. 1, 1998 08:02:00". The "root" flag is set to a false state in the "root" flag field 330 because Message 3 is not a conversation root and depends from Message 2, as shown in FIG. 4a. Also, a "dirty" flag for Message 3 is set to a true state in the "dirty" flag field 335, and a "place holder" flag is set to a false state in the "place holder" flag field 340.

Referring again to FIG. 6a, Message 4 is downloaded and the data fields 300, 305, 310, 315, 320, 325, 330, 335, and 340 within the database 39 are populated with message-related information associated with the message. The message-related information for a message entry associated with the message having a message ID "D" includes: EID "4", article number "100", title "Message 4", message ID "D", parent ID "A", and posted time "Jan. 1, 1998 08:03:00". The "root" flag is set to a false state in the "root" flag field 330 because Message 4 is not a conversation root and depends from Message 1, as shown in FIG. 4a. Also, a "dirty" flag for Message 4 is set to a true state in the "dirty" flag field 335, and a "place holder" flag is set to a false state in the "place holder" flag field 340.

After all of the messages have been downloaded and the data fields have been populated, the message manager 37 generates a conversation index for each message entry having a "dirty" flag set to a true state. Generation of a conversation index is described in greater detail below in connection with FIG. 12. The conversation index for a conversation root is generated first because other messages may depend on the conversation root and share a common preamble, which is based on the conversation index of the conversation root. For example, referring back to FIG. 4a, Message 2 102, Message 3 104, and Message 4 106, all depend from Message 1 100. Hence, referring to FIG. 4c, the preamble for each message consists of information derived from Message 1 having a message ID "A".

Referring again to FIG. 6a, in this example, the conversation root is Message 1 based on the fact that the "root" flag is set to a true state in the database 39. As a result, the conversation index for Message 1 is generated in a MAPI conversation threading scheme similar to the one shown for

15

Message 1 100a in FIG. 4c and previously described in connection with FIG. 5. After this conversation index is generated, the conversation indices for the reply messages, namely Message 2, Message 3, and Message 4, are generated. The reply messages maintain a conversation threading scheme similar to the structures 102a, 104a, and 106a illustrated in FIG. 4c.

After all of the conversation indices are generated, the conversation indices are stored in the local message store 38 in connection with the appropriate messages. Also, the "dirty" flags then are set to a false state to indicate that the conversation indices are set for the messages, as shown in FIG. 6b. The client-server session is completed and discontinued.

Advantageously, an exemplary embodiment of the present invention provides means for efficiently managing threaded conversations. A database is consulted during download operations to determine if message entries exist for downloaded messages, and if not, message entries are created for those messages that are not already entered in the database. The database is used to create and manage threaded conversations. Moreover, if any data changes with respect to the messages, such as a message becoming dirtied, the database is updated to reflect those changes so that information in the MAPI store remains in synchronization with server information. A conversation index is created for each message entry so that messages like those provided in news groups can operate in a MAPI format and be used in a software environment, like that of the "MICROSOFT OUTLOOK '98" program. Advantageously, the conversation index is updated only when necessary, thereby minimizing the amount of changes made in the MAPI store.

These features and advantages can be further appreciated by way of a second representative example. The second example is a more complex case in which a root message and reply messages are downloaded during several client-server sessions. This example is illustrated in FIGS. 7a-7j with continuing reference to FIGS. 3, 4b, and 4d. In this example, a connection is made between a client 20 (FIG. 3) and server 49 (FIG. 3). A user desires to check her download news messages and is typically prompted by the e-mail program module 36 (FIG. 3) to enter a password for access to the messages.

In this example, assume there are four client-server sessions. In a first client-server session, there is only one message, namely Message 7, on the server 49 to be downloaded to the client 20. As shown in FIG. 4b, Message 7 has a references field consisting of <D><E><F> and has a message ID "G". In a second client-server session, there is only one message, namely Message 8, to be downloaded from the server 49 to the client 20. In FIG. 4b, Message 8 has a references field consisting of <A><D><E> and has a message ID "H". In a third client-server session, there are three messages to be downloaded. Referring still to FIG. 4b, these messages include: Message 1 having no references in its reference field and having a message ID "A"; Message 4 having a references field consisting of <A> and having a message ID "D"; and Message 5 having a references field consisting of <A><D> and having a message ID "E". Finally, in a fourth client-server session, there is only one message to be downloaded, namely Message 6, which has a references field consisting of <A, D, E> and has a message ID "F".

Turning to FIG. 7a, in the first client-server session, the client 20 transmits a command to retrieve a message ID for a message. The server responds by transmitting to the client

16

a message ID "G". The client checks the database 39 to determine whether there are any message entries having the message ID "G". In this example, there are no message entries having this message ID. As a result, the associated message, in this case Message 7, is downloaded in the local message store 38.

In response to downloading the message, the data fields 300, 305, 310, 315, 320, 325, 330, 335, and 340 within the database 39 are populated with message-related information associated with the Message 7. This information includes: EID "1", article number "200", title "Message 7", message ID "G", parent ID "F", and posted time "Jan. 2, 1998 08:00:00". As previously mentioned in connection with FIGS. 6a and 6b, the parent ID is typically the last referenced message ID in the references field of the message. In this case, Message 7 has a references field consisting of <D><E><F> and the last referenced message ID is "F". Hence, the message having a message ID "F" is the parent ID for Message 7. The "root" flag is set to a false state in the "root" flag field 330 because Message 7 is not a conversation root and depends from Message 6, as shown in FIG. 4a. Also, a "dirty" flag for Message 7 is set to a true state in the "dirty" flag field 335. In addition, a "place holder" flag is set to a false state in the "place holder" flag field 340.

Due to the fact that Message 7 has references, a determination is made as to whether each referenced message ID matches a message ID in the message ID field 315 in the database. The determination is made by checking the last referenced message ID first and proceeding from right to left in the references field for the message. If there is a match, the message having the referenced message ID has already been downloaded. On the other hand, if there is not a match, the associated message has not been downloaded. Hence, a place holder is created for the referenced message ID. Any information about the non-downloaded message that can be obtained from the already downloaded message is stored in appropriate data fields in the database as an incomplete message entry.

In this example, the last referenced message ID is "F" for Message 7. This message ID is not already found in the database 39. Therefore, a place holder is created for the non-downloaded message. Based on the message-related information for Message 7 in the database, the place holder includes: EID "2", Article number "200", message ID "F", and parent ID "E". The parent ID is preferably determined by checking the next referenced message ID, from right to left, in the references field of Message 7. This method of determining the parent ID is generally reliable because there is a presumption that all of the messages are referenced in the references field and that the message IDs are presented in order. The "root" flag is set to a false state because the message has a parent ID. The "dirty" flag is set to a true state, and the "place holder" flag is set to a true state. This procedure is executed for each referenced message ID in the references field of the message.

In FIG. 7a, a place holder is created for message ID "E". The place holder includes: EID "3", Article number "200", message ID "E", and parent ID "D". The "root" flag is set to a false state; the "dirty" flag is set to a true state; and the "place holder" flag is set to a true state. In addition, a place holder is created for message ID "D". The place holder includes: EID "4", Article number "200", message ID "D", and no parent ID. The message manager determines that the message having message ID "D" is a conversation root because there are no more referenced messages. As a result, the "root" flag is set to a true state. In addition, the "dirty" flag is set to a true state, and the "place holder" flag is set to a true state.

17

After the message has been downloaded and the message-related information is populated in the database 39, the message manager 37 generates a conversation index for each message entry having a "dirty" flag set to a true state. The conversation root preferably is generated first. In this example, the conversation root is the message having the message ID "D" as noted by the "root" flag in the database 39. The conversation index 116a (FIG. 4d) is generated for the message having the message ID "D". Once the conversation index for the conversation root is generated, the message manager 37 methodically determines which conversation index to generate next based on a message entry's dependency on the conversation root. In other words, the message manager 37 preferably checks the parent ID field 320 to determine which message entries directly depend from the conversation root and generates the conversation index for those message entries. Based on those message entries, the message manager 37 checks the parent ID field 320 for the next message entries and generates the conversation index for the next message entries. This process continues until all conversation indices have been generated for message entries having a "dirty" flag set to a true state.

In this example, the message entry having the message ID "E" depends directly from message ID "D". As a result, the conversation index 118a (FIG. 4d) for message ID "E" is generated. Next, the message entry having the message ID "F" depends directly from message ID "E". So, the conversation index 120a (FIG. 4d) for message ID "F" is generated. Finally, the message entry having the message ID "G" depends directly from message ID "F". So, the conversation index 122a (FIG. 4d) for message ID "E" is generated.

After all of the conversation indices are generated, the conversation indices are stored in the local message store 38 in connection with the appropriate messages. Also, the "dirty" flags then are set to a false state to indicate that the conversation indices are set for the messages, as shown in FIG. 7b.

Referring to FIG. 7c, in the second client-server session, there is only one message, namely Message 8, to be downloaded from the server 49 to the client 20. In FIG. 4b, Message 8 has a references field consisting of <A><D><E> and has a message ID "H". The message is downloaded and data fields are populated in a similar manner as previously described in connection with FIG. 7a. However, in this example, the message entry includes: EID "5", article number "200", title "Message 8", message ID "H", parent ID "E", and posted time "Jan. 3, 1998 08:00:00". A "dirty" flag is set to a true state for the message, and a "place holder" flag is set to a false state for the message entry. As noted, the parent ID is the last referenced message ID in the references field.

As previously mentioned, the message manager checks each referenced message ID to determine whether there is a match in the database 39. In this example, the last referenced message ID "E" is already located in the database 39, thereby indicating that the message entry is either a place holder or the message has already been downloaded. The message manager 37 checks the next referenced message ID "D". Message ID "D" is also already located in the database 39. The message manager 37 then checks the next referenced message ID "A". Message ID "A" is not found in the database, and therefore, a place holder is created for message ID "A". The place holder includes: EID "6", Article number "200", message ID "A", and no parent ID. The message manager 37 determines that the message having message ID "A" is a conversation root because there are no more referenced messages. As a result, the "root" flag is set

18

to a true state. In addition, the "dirty" flag is set to a true state, and the "place holder" flag is set to a true state.

As noted in the references field for Message 8, the referenced message ID "D" depends from the referenced message ID "A". Based on this information, the message manager determines that the message having the message ID "D" is not a conversation root. As a result, the place holder for message ID "D" is updated. Specifically, the parent ID field 320 for message ID "D" is updated to indicate message ID "A" as the parent ID. Also, the "root" flag field 330 is updated by changing the "root" flag from a true state to a false state for message ID "D". Finally, all message entries that depend from the conversation root having the message ID "A" are updated by changing the "dirty" flags from a false state to a true state, as shown in FIG. 7c. The true state of the "dirty" flag serves as an indicator that either the conversation index has not been generated or the conversation index must be updated.

Next, the message manager 37 generates a conversation index for each message entry having a "dirty" flag set to a true state. In this example, the conversation root is the message having a message ID "A" based on the fact that the "root" flag is set to a true state in the database 39. The conversation index 110a (FIG. 4d) is generated for the message having a message ID "A". After this conversation index is generated, the conversation indices are generated for all the messages that depend from the conversation root. In this example, a conversation index is generated in a manner previously described in connection with FIG. 7a for all of the message entries.

After all of the conversation indices are generated, the conversation indices are stored in the local message store 38 in connection with the appropriate messages. Also, the "dirty" flags then are set to a false state to indicate that the conversation indices are set for the messages, as shown in FIG. 7d.

In a third client-server session, there are three messages to be downloaded. Referring to FIG. 4b, these messages include: Message 1 having no references in its reference field and having a message ID "A"; Message 4 having a references field consisting of <A> and having a message ID "D"; and Message 5 having a references field consisting of <A><D> and having a message ID "E".

Referring to FIG. 7e, the message manager 37 determines that the message ID "A" is located in the database and that the message entry for message ID "A" is a place holder. Based on this information, Message 1 is downloaded to the local message store and the appropriate data fields are populated to provide a complete message entry. Specifically, the title field 310 is populated with the title "Message 1" and the posted time field 325 is populated with the date and time sent "Jan. 4, 1998 08:00:00". A "dirty" flag is set to a true state for the message, and a "place holder" flag is set to a false state for the message entry. In addition, all message entries that depend from the conversation root having the message ID "A" are updated by changing the "dirty" flags from a false state to a true state, as shown in FIG. 7e.

Next, turning to FIG. 7f, the message manager 37 retrieves the next message ID, namely message ID "D", from the server 49 and determines that the message ID "D" is located in the database 39. The message manager 37 further determines that the message entry for message ID "D" is a place holder. Based on this information, Message 4 is downloaded to the local message store 38, and in the database 39, the title field 310 is populated with the title "Message 4" and the posted time field 325 is populated with

19

the date and time sent "Jan. 4, 1998 08:01:00". The "dirty" flag is already set to a true state for the message because the message depends from the conversation root Message 1, which requires its conversation index to be regenerated. The "place holder" flag is changed from a true state to a false state for the message entry.

Next, turning to FIG. 7g, the message manager 37 retrieves the next message ID, namely message ID "E", from the server 49 and determines that the message ID "E" is located in the database 39. The message manager 37 further determines that the message entry for message ID "E" is a place holder. Based on this information, Message 5 is downloaded to the local message store 38, and in the database 39, the title field 310 is populated with the title "Message 5" and the posted time field 325 is populated with the date and time sent "Jan. 4, 1998 08:02:00". The "dirty" flag is already set to a true state for the message because the message depends from the Message 4, which requires its conversation index to be regenerated. The "place holder" flag is changed from a true state to a false state for the message entry.

After all of the messages have been downloaded from the server 49, the message manager 37 generates a conversation index for each message entry having a "dirty" flag set to a true state. The conversation indices are then stored in the local message store 38 in connection with the appropriate messages. Also, the "dirty" flags are set to a false state to indicate that the conversation indices are set for the messages, as shown in FIG. 7h.

Finally, in the fourth client-server session, there is only one message to be downloaded, namely Message 6, which has a references field consisting of <A><D><E> and has a message ID "F". The message manager performs a similar procedure as previously described in connection with FIGS. 7a-7h.

Referring to FIG. 7i, Message 6 is downloaded to the local message store 38 and the appropriate data fields in the database 39 are populated. Specifically, the title field 310 is populated with the title "Message 6" and the posted time field 325 is populated with the date and time sent "Jan. 4, 1998 08:03:00". The "dirty" flag is set to a true state for the message, and the "place holder" flag is changed from a true state to a false state for the message entry. In addition, any messages that depend from Message 6 is updated by changing the "dirty" flag from a false state to a true state. In this example, Message 7 is the only message that depends from Message 6. As a result, the "dirty" flag is changed from a false state to a true state.

The message manager 37 then generates a conversation index for each message entry having a "dirty" flag set to a true state, which in this case includes Message 6 and Message 7. The conversation indices are then stored in the local message store 38 in connection with the appropriate messages. Also, the "dirty" flags are set to a false state to indicate that the conversation indices are set for the messages, as shown in FIG. 7j.

In view of the foregoing, the present invention effectively and efficiently keeps track of messages that are downloaded from a server to a client and maintains the message order of a threaded conversation. Advantageously, the present invention provides place holders for messages so that missing messages do not ruin the message order.

With continuing reference to FIGS. 1-7, FIG. 8 is a flow diagram illustrating an exemplary method of operation of the present invention. Those skilled in the art will appreciate that this exemplary method of operation is carried out by a

20

client using an application program 36 (FIG. 2), such as the "MICROSOFT OUTLOOK '98" program. The "MICROSOFT OUTLOOK '98" program has a message manager program module 37 (FIG. 3), which is operative to communicate with the remote mail server 49 (FIG. 3). In this embodiment, the client, the remote mail server, and the local message store implement an Internet protocol and operate in an on-line mode.

At start step 500, the exemplary program module is initialized and a connection is made between the client and the server, typically via a modem, for communication. Also, at start step 500, a log-in procedure typically occurs, where a user is prompted to enter a password, and the password is verified by the program module.

Next, in step 505, a command to retrieve a message ID is transmitted from the client to the server. Each message preferably has a message identifier or message ID to identify the message. In response to the command, in step 510, the client retrieves a message ID from the server. Next, the message ID is compared to each message entry in a client-based database, in step 510. The database preferably contains message-related information for news messages. This message-related information is maintained in the database particularly in connection with generating a conversation index for conversation threading.

A central inquiry is made, in step 520, as to whether the message ID retrieved from the server matches any of the Message IDs in the database. If there is a match, the "YES" branch is followed to step 530; otherwise, the "NO" branch is followed to step 525. In step 530, another inquiry is made as to whether the message entry is a place holder.

To determine whether a message entry is a place holder, the message manager checks the state of a "place holder" flag for the message. If the "place holder" flag is set to a false state, the message entry is not a place holder, and the "NO" branch then is followed to step 535. In step 535, a determination is made as to whether there are any additional messages located on the server. If there are additional messages located on the server, the "YES" branch is followed to step 510, in which case, the next message ID is retrieved; otherwise, the "NO" branch is followed to the end step 575.

If the "place holder" flag is set to a true state, the message entry is a place holder, and the "YES" branch is followed to step 525. In step 525, download operations are performed to download the message from the server to the client. The process of downloading a message and setting data fields in the database are described in greater detail below in connection with FIG. 9.

After download operations are performed, in step 527, message entries are updated, if necessary, in the database. For example, parent ID fields may be updated to reflect changes in conversation roots, "dirty" flags may change states, "root" flags may change states, and so forth.

After the message entries have been updated, in step 540, a determination is made as to whether there are any additional messages located on the server. If there are additional messages located on the server, the "YES" branch is followed to step 510, in which case, the next message ID is retrieved; otherwise, the "NO" branch is followed to step 545.

A determination is made, in step 545, as to whether there have been any conversation roots downloaded to the client. If a conversation root has been downloaded, the "YES" branch is followed to step 555; otherwise, the "NO" branch is followed to step 550. In step 555, a conversation index is

generated for the conversation root, as previously described in connection with FIGS. 4-7. Next, in step 570, a conversation index is generated for all reply messages that depend on the conversation root. If a reply message does not have a conversation root or no conversation roots have been downloaded, in step 550, a conversation index is generated based on the parent ID.

Once all conversation indices have been generated, in step 560, the conversation indices are stored in association with their respective downloaded messages in a local message store, such as a MAPI store. Next, all "dirty" flags are set to a false state to indicate that a conversation index has been generated for the respective message. The process terminates at the end step 575.

Now turning to FIG. 9, a flow diagram illustrating an exemplary process of downloading a message from the server to the client is described. Specifically, steps 600-690 represent the process of downloading a message and setting data fields in a database as described in connection with step 525 of FIG. 8.

In step 600, a message is downloaded from the server to the MAPI store located at the client. Next, in step 605, data fields in the database are populated with message-related information, such as an article number, a title, and a message ID for the message. The process of populating data fields is described in greater detail below in connection with FIGS. 10a and 10b. Specifically, FIG. 10a illustrates the population of data fields for a message that does not have a matching message ID in the database, and FIG. 10b illustrates the population of data fields for a message that has a place holder in the database.

Referring still to FIG. 9, once data fields are populated with the message-related information, a "dirty" flag is set to a true state in step 610. As previously mentioned in connection with FIGS. 6a and 6b, the true state of the "dirty" flag is indicative of a message that does not have a conversation index set in the local message store. The true state of the "dirty" flag is also indicative of the message being "dirtied" because a conversation index of another message from which the message depends has not been set or has become "dirtied".

In step 615, a "place holder" flag is set to a false state. The false state of the "place holder" flag is indicative of the message being downloaded and the message-related information for the downloaded message being populated in the database.

Next, the references field of the message is checked for message IDs, in step 620. Each message has a references field that can include a single message ID, a group of message IDs, or no reference at all. In the case where the message has no reference, the message is a conversation root or root message. In other words, the message with no reference is the original message having no links to other messages. In the case where the message has at least one message ID, the message is a reply or nested message and is linked to any messages referenced in the references field.

Next, a determination is made, in step 625, as to whether there are any message IDs in the references field of the downloaded message. If so, the "YES" branch is followed to step 635, in which case a "root" flag is set to a false state. The false state of the "root" flag is indicative of the message being a reply message and not a conversation root. If there is no message ID in the references field, the "NO" branch is followed to step 630, in which case the "root" flag is set to a true state. The true state of the "root" flag is indicative of the message being a root message or conversation root. After step 630, the process branches to step 527 (FIG. 8).

In the case where the message has at least one reference, in step 640, the last referenced Message ID in the references field is checked. This referenced message is considered the parent of the downloaded message. Next, in step 645, the parent ID field is populated with the last referenced message ID.

Based on this referenced message ID, a determination is made, in step 650, as to whether the referenced message ID matches any of the message IDs in the database. If there is a match, the "YES" branch is followed to step 655; otherwise, the "NO" branch is followed to step 665. In step 655, a central inquiry is made as to whether there are additional references in the references field of the message. If so, the "YES" branch is followed to step 660, in which case the next referenced message ID in the references field is checked. It will be understood that the next referenced message ID is the reference located to the immediate left of the last referenced message ID in a tree structure conversation threading scheme.

Once the next referenced message ID is checked, the same determination is made, in step 650, as to whether the next referenced message ID matches any of the message IDs in the message ID field in the database. If there are not any additional references in the references field of the message, the "NO" branch is followed to step 527 (FIG. 8).

In step 665, a place holder is created for the referenced message ID when there is not a match for the message ID in the database. The process of creating a place holder is described in greater detail below in connection with FIG. 11. Next, a determination is made, in step 670, as to whether there are additional references in the references field of the message. If so, the "YES" branch is followed to step 680, in which case the "root" flag for the message is set to a false state; otherwise, the "NO" branch is followed to step 675, in which case the "root" flag for the message is set to a true state. After step 675, the process branches to step 527 (FIG. 8).

Upon completion of step 680, the next referenced message ID in the references field is checked, in step 685. This referenced message is considered the parent of the non-downloaded message having the place holder. Consequently, in step 690, the parent ID field is populated with the next referenced message ID. Based on the next referenced message ID, a determination is made, in step 650, as to whether the referenced message ID matches any of the message IDs in the database. This entire process, steps 600-690, is performed for each message on the server.

With continuing reference to FIG. 9, FIGS. 10a and 10b, collectively referred to as FIG. 10, are flow diagrams illustrating an exemplary process of populating data fields in accordance with an exemplary embodiment of the present invention. Referring to FIG. 10a, the population of data fields for a message that does not have a matching message ID in the database is described, where steps 700-725 represent this process as described in connection with step 605 of FIG. 9. In step 700, an entry identifier or ID is populated in an entry ID field in the database. Next, an article number for the message is populated in an article number field in the database, in step 705. A title for the message is then populated, in step 710, in a title field in the database. In step 715, a message identifier is populated in a message ID field in the database. Next, in step 720, a parent identifier or ID is populated in a parent ID field in the database. Finally, a posted or sent time and date for the message is populated, in step 725, in a posted time field in the database.

Turning to FIG. 10b, a flow diagram illustrating the population of data fields for a message that has a place

holder in the database is described, where steps 730 and 735 represent this process as described in connection with step 605 of FIG. 9. In the case where a place holder is in the database, specific message-related information has already been placed in their appropriate fields. As previously mentioned in connection with FIG. 9, the process of creating a place holder is described below in FIG. 11. Hence, the remaining information is stored in the database after the message has been downloaded from the server. Specifically, in step 730, a title for the message is populated in the title field in the database. Finally, a posted time and date for the message is populated, in step 735, in a posted time field in the database.

With continuing reference to FIGS. 1-10, FIG. 11 is a flow diagram illustrating an exemplary process for creating a place holder in accordance with an exemplary embodiment of the present invention. Specifically, steps 800-825 represent this process as described in connection with step 665 of FIG. 9. In step 800, an entry identifier or ID is populated in an entry ID field in the database. Next, an article number for the message is populated in an article number field in the database, in step 805. A message identifier or ID is populated, in step 810, in a message ID field in the database. Next, in step 815, a parent identifier or ID is populated in a parent ID field in the database. A "dirty" flag for the message is set to a true state, in step 820. Finally, a "place holder" flag for the message is set to a true state, in step 825, thereby indicating that the message entry is a place holder.

Referring to FIG. 12, a flow diagram illustrating an exemplary process for generating a conversation index based on message-related information in a client-based database is described. In the start step 900, all of the messages in a client-server session have been downloaded and the appropriate fields in the database have been populated. In addition, at the start step 900, assume a message is selected having a references field containing only one message ID, thereby indicating that the message is a reply message and not a root message.

To generate a conversation index for the message, in step 905, the message manager places a value of "1" in a first field of the conversation index. Typically, the first field is unused in MAPI format-sensitive applications. Consequently, a value of "1" is used as a default value in the first field. Next, in step 910, the message manager obtains from the database a time and date for the root message. The message ID for the root message is also obtained from the database, in step 915. Next, in step 920, the message manager obtains from the database a time and date for the reply message. Once the information in steps 910, 915, and 920 have been obtained from the database, in step 925, the exemplary program module stores the time and date of the root message in a second field of the conversation index. Next, the message manager generates, in step 930, a unique identifier for the root message based on the message ID of the root message obtained from the database. The unique identifier is generated by using a "hashing" process to convert the arbitrary length of the message ID to the fixed length of a unique identification field. The unique identifier then is stored in the unique identification field or a third field of the conversation index, in step 935.

Next, in step 940, the message manager calculates the difference between the time and date of the reply message and the time and date of the root message to produce a difference "delta" for the reply message. This difference "delta" is then stored, in step 945, in a fourth field. The delta of the conversation index represents a child of the conversation. Moreover, a delta preferably is generated for each

child of the conversation when there are additional references in the reference field. The process of generating a conversation index terminates at the end step 950.

In summary, it will be understood that the present invention provides a system for managing messages communicated within a client-server architecture, such as a distributed computing environment represented by a client and a server. The present invention uses a database, stored at the client, to maintain a central archive of message-related information to support conversation threading operations between the client and the server. Communications with a message server are facilitated by accessing a client-based database representing a central archive of message-related information. Archived information is accessed in the client-based database during typical message communications operations, such as message download operations. This information is then used to convert messages from a tree structure conversation threading scheme to a MAPI format for use in a MAPI store.

The invention may conveniently be implemented in one or more program modules that are based upon and implement the features illustrated in FIGS. 3-11. No particular programming language has been described for carrying out the various procedures described above because it is considered that the operations, steps, and procedures described above and illustrated in the accompanying drawings are sufficiently disclosed to permit one of ordinary skill in the art to practice the present invention. Moreover, there are many computers and operating systems which may be used in practicing the present invention and therefore no detailed computer program could be provided which would be applicable to all of these many different systems. Each user of a particular computer will be aware of the language and tools which are most useful for that user's needs and purposes. In addition, although the invention was described in the context of a e-mail client, mail server, and message store, that implement one of a variety of Internet protocols, those skilled in the art will appreciate that the invention is applicable to remote servers that include other types of data stores, message stores, and file stores, and in other operating environments.

Alternative embodiments will become apparent to those skilled in the art to which the present invention pertains without departing from its spirit and scope. Accordingly, the scope of the present invention is defined by the appended claims rather than the foregoing description.

What is claimed is:

1. In a distributed computer system including a server and a client, the client including a local message store and a database, a method for managing conversation threads based on message-related information for a message having a message identifier and a references field, the message-related information being stored in the database, comprising the steps of:

- (a) during a client-server session, retrieving from the server the message-related information corresponding to the message;
- (b) based on the message-related information, determining whether the message has been previously downloaded from the server to the local message store located at the client;
- (c) in response to determining that the message has not been previously downloaded from the server to the local message store,
  - i. downloading the message from the server to the local message store, and

25

- ii. populating a plurality of data fields in the database with the message-related information;
  - (d) determining whether the references field of the message is empty;
  - (e) in response to determining that the references field of the message is empty, providing an indication in the database that the message is a conversation root;
  - (f) providing an indication in the database that a conversation index has not been determined for the message;
  - (g) repeating the steps (a) through (f) for each remaining message on the server; and
  - (h) generating, based on the conversation root, the conversation index for each message downloaded from the server to the local message store.
2. The method of claim 1, wherein the message-related information comprises the message identifier for identifying the message, an article number for identifying the number of the article for the message, a title for identifying the title of the message, and a parent identifier for identifying the parent of the message.
3. The method of claim 2, wherein the message-related information further comprises a posted time for indicating a time and date that the message is posted to a news server.
4. The method of claim 3, wherein the data fields of the database comprise a message identification field for the message identifier, an article number field for the article number, a title field for storing the title of the message, a parent identification field for storing the parent identifier for the message, and a posted time field for storing the time and date of the message.
5. The method of claim 4, wherein the step of populating the data fields in the database with the message-related information further comprises the steps of:
- populating the message identification field with the message identifier;
  - populating the article number field with the article number;
  - populating the title field with the title of the message;
  - populating the parent identification field with the parent identifier for the message; and
  - populating the posted time field with the time and date of the message.
6. The method of claim 1, wherein the step of providing an indication in the database that a conversation index has not been determined for the message comprises setting a "dirty" flag to a true state in the database.
7. The method of claim 1, wherein the step of generating, based on the conversation root, the conversation index for each message downloaded from the server to the local message store comprises:
- creating a preamble based on the conversation root; and
  - if the message is not the conversation root, creating a reply message field for maintaining a difference delta defining a difference between a sent time of the message and a sent time of the conversation root.
8. The method of claim 7, wherein the step of creating the preamble based on the conversation root comprises:
- placing a value of one in a first field of the preamble;
  - storing the sent time of the conversation root in a second field of the preamble; and
  - generating a unique identifier based on the message identifier of the conversation root; and
  - storing the unique identifier in a third field of the preamble.

26

9. The method of claim 1, wherein after generating the conversation index for each downloaded message, the method further comprises the step of providing an indication in the database that the conversation index has been determined for the message.

10. The method of 9, wherein the step of providing an indication in the database that the conversation index has been determined for the message comprises setting a "dirty" flag to a false state in the database.

11. The method of 1, wherein after generating the conversation index for each downloaded message, the method further comprising the step of storing the conversation index for the downloaded message in the local message store.

12. The method of 1, wherein the step of determining whether the message has been downloaded from the server to the local message store comprises comparing the message identifier for the message to at least one message identifier in the database and determining whether the message identifier for the message matches the one message identifier in the database.

13. A method for generating a conversation index based on message-related information for a reply message, the reply message arranged in a news conversation threading structure and including a message identifier and a reference, the reference including a message identifier for a root message, the message-related information stored in a database, located at the client, and including the message identifier and a time and date for the root message and the reply message, the conversation index supporting conversation threading so that the root message and the reply message are grouped in a MAPI format for use in a MAPI format-sensitive application, the conversation index comprising fields arranged in the MAPI format, the method comprising the steps of:

- (a) placing a value in a first field of the conversation index;
- (b) obtaining from the database:
  - i. the time and date for the root message,
  - ii. the message identifier for the root message, and
  - iii. the time and date for the reply message;
- (c) storing the time and date of the root message in a second field of the conversation index;
- (d) generating a unique identifier based on the message identifier of the root message;
- (e) storing the unique identifier in a third field of the conversation index;
- (f) determining a difference between the time and date of the reply message and the time and date of the root message; and
- (g) storing the difference between the time and date of the reply message and the time and date of the root message in a fourth field of the conversation index, the conversation index representing the reply message in the MAPI format for use in the MAPI format-sensitive application.

14. The method of claim 13, wherein the reply message has an additional reference containing a message identifier for a second reply message, the message identifier for the second reply message and a time and date for the second reply message being stored in the database; and

- wherein the method further comprises:
  - consulting the database to obtain the time and date for the second reply message;
  - determining a difference between the time and date of the second reply message and the time and date of the root message; and

27

storing the difference between the time and date of the second reply message and the time and date of the root message in a fifth field of the conversation index.

15. The method of claim 13, wherein placing a value in a first field of the conversation index comprises placing a value of "1" in the first field of the conversation index.

16. The method of claim 13, wherein the step of generating a unique identifier based on the message identifier of the root message comprises performing a hashing process to convert the length of the message identifier to a fixed length.

17. The method of claim 13, wherein the reply message has additional references each including a message identifier for an additional message and a time and date for the additional message, the method further comprising for each additional reference, determining a difference delta, delta defining the difference between the time and date of the additional message and the time and date of the root message, and for each additional reference, storing the difference delta in an additional field of the conversation index.

18. A method for creating, in a database located at a client, a place holder from a reply message that has been downloaded in the database from a server, the reply message including a references field for hosting a reference comprising a message identifier for a second message, the method comprising the steps of:

- (a) determining whether the references field of the reply message includes the reference comprising the message identifier for the second message;
- (b) in response to determining that the reference includes the message identifier for the second message, determining whether the message identifier for the second message is located in a message identification field in the database, the message identification field storing message identifiers for messages;
- (c) in response to determining that the message identifier for the second message is not located in the message identification field, downloading into the database a message entry from the server, comprising the message identifier for the second message; and
- (d) providing an indication in the database that the message entry is a place holder.

19. The method of claim 18, wherein further in response to determining that the reference includes the message identifier for the second message, storing the message identifier for the second message as a parent identifier for the reply message.

20. The method of claim 18, further comprising the step of providing an indication in the database that a conversation index has not been generated.

21. The method of claim 20, wherein the step of providing an indication in the database that a conversation index has not been generated comprises setting a "dirty" flag to a true state in the database.

22. The method of claim 18, further comprising the steps of:

- determining whether there is an additional reference in the references field for the reply message, the additional reference including a message identifier for an additional message;
- in response to determining that there is no additional reference in the references field for the reply message, providing an indication in the database that the second message is a root message; and
- generating, based on the second message, a conversation index for the second message and the reply message.

28

23. The method of claim 22, further comprising the steps of:

in response to determining that there is the additional reference for the message reply, providing an indication in the database that the second message is not the root message;

storing the message identifier for the additional message as the parent identifier for the second message;

determining whether the message identifier for the additional message is located in the message identification field in the database;

in response to determining that the message identifier for the additional message is not located in the message identification field, downloading into the database a second message entry comprising the message identifier for the additional message from the server; and

providing an indication in the database that the second message entry is a place holder.

24. The method of claim 23, wherein the step of providing an indication in the database that the second message is not the root message comprises setting a "root" flag to a false state in the database.

25. The method of claim 22, wherein the step of providing an indication in the database that the second message is the root message comprises setting a "root" flag to a true state in the database.

26. The method of claim 22, wherein the step of generating, based on the second message, a conversation index for the second message comprises creating a preamble based on the second message comprising the steps of:

placing a value in a first field of the conversation index; selecting an arbitrary time and date for the second message;

storing the arbitrary time and date of the second message in a second field of the conversation index; and

based on the message identifier of the second message, storing a unique identifier in a third field of the conversation index.

27. The method of claim 26, wherein the step of generating, based on the second message, a conversation index for the reply message comprises the steps of:

determining a difference between a time and date of the reply message and the arbitrary time and date of the second message; and

storing the difference between the time and date of the reply message and the arbitrary time and date of the second message in a fourth field of the conversation index.

28. The method of claim 18, wherein the step of providing an indication in the database that the message entry is a place holder comprises setting a "place holder" flag to a true state in the database.

29. A method for generating a conversation index based on message-related information for a root message, the root message arranged in a news conversation threading structure and including a message identifier, the message-related information stored in a database, located at the client, and including the message identifier and a time and date for the root message, the conversation index supporting conversation threading so that the root message is converted to a MAPI format for use in a MAPI format-sensitive application, the conversation index comprising fields arranged in the MAPI format, the method comprising the steps of:

29

- (a) placing a value in a first field of the conversation index;
- (b) obtaining from the database:
  - i. the time and date for the root message, and
  - ii. the message identifier for the root message;
- (c) storing the time and date of the root message in a second field of the conversation index;
- (d) generating a unique identifier based on the message identifier of the root message; and

30

- (e) storing the unique identifier in a third field of the conversation index, the conversation index representing the root message in the MAPI format for use in the MAPI format-sensitive application.
30. The method of claim 29, wherein placing a value in a first field of the conversation index comprises placing a value of "1" in the first field of the conversation index.

\* \* \* \* \*